Efficient Transmission and Reconstruction of Dependent Data Streams via Edge Sampling

Joel Wolfrath and Abhishek Chandra
Department of Computer Science and Engineering
University of Minnesota, Minneapolis, USA
Email: {wolfr046, chandra}@umn.edu

***Abstract*—Data stream processing is an increasingly important topic due to the prevalence of smart devices and the demand for real-time analytics. Geo-distributed streaming systems, where cloud-based queries utilize data streams from multiple distributed devices, face challenges since wide-area network (WAN) bandwidth is often scarce or expensive. Edge computing allows us to address these bandwidth costs by utilizing resources close to the devices, e.g. to perform sampling over the incoming data streams, which trades downstream query accuracy to reduce the overall transmission cost. In this paper, we leverage the fact that correlations between data streams may exist across devices located in the same geographical region. Using this insight, we develop a hybrid edge-cloud system which systematically trades off between sampling at the edge and estimation of missing values in the cloud to reduce traffic over the WAN. We present an optimization framework which computes sample sizes at the edge and systematically bounds the number of samples we can estimate in the cloud given the strength of the correlation between streams. Our evaluation with three real-world datasets shows that compared to existing sampling techniques, our system could provide comparable error rates over multiple aggregate queries while reducing WAN traffic by 27-42%.**

***Index Terms*—Stream processing, edge computing, big data, approximate computing**

I. Introduction

Real-time analytics and data-driven insights continue to play a pivotal role in online applications and business operations. Organizations require efficient query mechanisms to handle increasingly large datasets and the ubiquity of smart devices. Data streaming applications in particular continue to grow at an unprecedented rate, with projections suggesting there will be more than 75 billion connected devices by the year 2025 [1]. This includes personal devices, smart home components, smart cities, retail environments and industrial settings. This enormous growth in data generated at the edge has driven much research into efficient data transfer, given the scarcity and expense associated with transferring data over the wide-area network (WAN) [2]–[4]. Recent strides in edge computing provide a mechanism for addressing these constraints by leveraging computation close to the data-generating devices, allowing us to make optimizations and reduce cost. While we focus on minimizing network traffic, edge resources can address many other objectives, including maximizing throughput or minimizing end-to-end latency.

Approximation techniques are often used to trade downstream query accuracy for a reduction in traffic over the WAN. Sampling at the network edge is commonly used to reduce transmissions without introducing substantial error in downstream queries [5]–[7]. Sampling also allows us to bound the amount of error introduced in many kinds of queries—an important property for quantifying worst-case performance in production deployments. When data arrives in the cloud, additional samples may be imputed or estimated using a model [8], [9]. These techniques can be effective in practice, but existing sampling algorithms focus on the properties of individual streams and models fail to provide performance guarantees when making inferences out-of-sample.

In this work, we propose a novel sampling algorithm for addressing WAN scarcity which exploits similarities in data streams collected by multiple localized devices in a geo-distributed environment. Prior research suggests that devices located in the same geographical region may produce data streams that are correlated or exhibit some kind of dependency [10]–[12]. These correlations arise naturally if the sensors[1] are recording the same phenomenon in a geographic region (e.g. temperature), or if there is a correlation between different sensors due to external factors (e.g. human occupancy patterns in a building or neighborhood). We identify and leverage these dependencies in real time to improve sampling and modeling decisions. Identifying these dependencies at the edge allows us to explicitly control imputation accuracy, since we have access to all of the data at the edge prior to sampling. We make the following research contributions:

- We develop a hybrid edge-cloud streaming system which addresses WAN scarcity by leveraging dependencies between multiple streams in real time (without any offline profiling).
- We present a novel sampling algorithm and associated optimization problem for computing the number of samples to send over the network and impute in the cloud, given the dependence structure. We also provide explicit bounds on the error introduced without making any strong assumptions about the underlying data distribution.
- We evaluate our system across a variety of real-world datasets using a Storm-based implementation[2] and examine its efficacy and its sensitivity to a variety of tuning parameters.

[1]We use the terms ”device” and ”sensor” interchangeably in this work.
[2]https://github.com/jswolfrath/edge-approx.

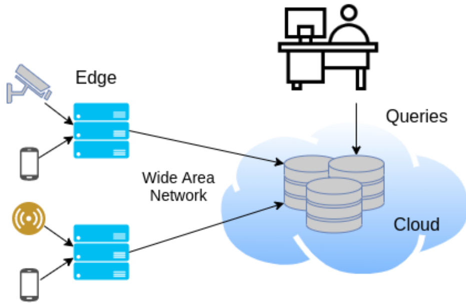


Fig. 1: A geo-distributed streaming system consisting of four data-generating devices. Data is sent to two local edge nodes prior to being transferred over the WAN and persisted in the cloud.

II. Preliminaries

To motivate our problem, we consider applications that are inherently latency sensitive and potentially bandwidth constrained. For example, wind turbines and smart trains can be equipped with IoT devices that generate data and send it to a centralized location [13], [14]. This data is used to identify failures in real time, resulting in timely repairs and substantial cost savings. Furthermore, monitoring the performance output of wind turbines is important for estimating revenue in real time [15]. Since trains and wind turbines are often in rural areas, the available bandwidth will be scarce. Urban settings can also benefit from bandwidth reduction techniques. Smart cities can have a very large number of sensors generating data simultaneously [16], [17], which increases the traffic over the WAN. These applications highlight the importance of efficient bandwidth usage in both rural and urban settings.

A. Problem Statement

We consider a three-tiered distributed streaming framework consisting of data-generating devices, edge nodes for local processing of this data, and a destination cloud data center where the data is finally persisted. In this work, we broadly define edge nodes to include routers, base stations, or dedicated edge servers. Figure 1 shows an example of this system topology. Streaming data typically consists of tuples and an associated timestamp. The key property is that the stream is unbounded, which implies the data observed at the edge is *transient* in nature. In this work, we assume a tumbling window model, where data is aggregated and processed as mini-batches across fixed periods of time. Selecting a windowing method and aggregation duration is an active area of research which heavily depends on the application and its tolerance for delay [18], [19].

When streaming data is persisted in the cloud, it can be represented in a variety of ways, ranging from storing simple point estimates (e.g. means or sums) to total reconstructions of the original streams. In this work, *our objective is to persist samples that minimize error when computing multiple aggregate functions*, such as counts, averages, standard deviations, and order statistics (e.g. minimum, maximum, or median). These aggregates allow us to preserve properties of the underlying data distribution without requiring a full reconstruction of the streams. The error minimization is subject to a budget constraint which dictates the amount of data we are allowed to send over the WAN in each window. Alternatively, the problem can be viewed as attempting to minimize the data transfer cost, subject to a specified tolerance for error.

B. Stream Sampling

In general, streaming high-frequency data over the WAN may be infeasible due to scarcity or high cost. This motivates sampling schemes at the edge which send a subset of the data while providing a bound on the error associated with downstream queries. To ensure a fixed transmission cost, one could select a simple random sample (SRS) of the data points observed in a tumbling window and forward those points to the cloud. However, in practice, an inbound data stream may be composed of several heterogeneous *sub-streams*, each with their own statistical properties [20]. This heterogeneity justifies the use of more sophisticated sampling techniques at the edge.

When each stream has distinct statistical properties or observed frequencies, *stratified sampling* may be preferable to a SRS [20], [21]. Widely used in practice [5], [22], this technique has the desirable property of ensuring each stratum of a partitioned dataset is represented in the sample. Some systems use stratification methods that leverage properties of the underlying data. For example, the S-VOILA system stratifies samples by assigning more samples to streams with a high degree of variability and fewer samples to streams with less variability [23]. These stream sampling techniques provide a mechanism for streaming applications to trade downstream query accuracy for a reduction in the amount of data sent over the network.

C. Imputation Strategies

One way to handling missing values in a dataset is to *impute* them, i.e., replace them with some kind of point estimate. It is common to replace missing values with a statistic (e.g. the mean) or estimate them using a model (e.g. linear regression) which is estimated using the available data. When sampling is performed at the edge, the cloud will necessarily receive a partial dataset which does not contain samples for every point in time. We propose a careful use of these imputation techniques to estimate a subset of the missing values while controlling for error. We leverage the fact that statistics or models used for imputation can be estimated at the edge, where more data is available to us as compared to the cloud, which can only work with the downsampled data.

III. Proposed Approach

A. Overview

We propose an edge-cloud framework for data streaming that combines edge-side sampling with cloud-based imputation of missing data values. We leverage edge resources to sample

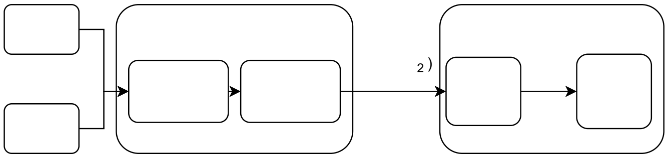


Fig. 2: Proposed edge-cloud streaming framework.

multiple incoming data streams and estimate dependencies between them. The edge also performs optimization to determine which samples and models to send over the network. The cloud uses these samples and models to impute missing data values. This framework uses the insight that streams produced by individual devices in the same geographical region may be correlated [10]–[12].

For example, this is expected when devices in the same sensor network generate streaming data [24], [25]. Mazhar *et al.* show that smart-home IoT traffic is often directly correlated with human activity patterns [11]. Even devices in a smart city that measure different quantities could produce correlated streams [10], [24], [26]. The real-world datasets we use for evaluation also substantiate this claim (see section V for details). Identifying these dependencies provides an opportunity for exploiting them for efficient data transfer.

To illustrate the opportunity, consider two data generating devices, D_1 and D_2, which are located in the same geographical region and forward their data to a single edge node. The local edge node caches the inbound streaming data for the duration of the tumbling window and then selects a subset of the received data to be forwarded to the cloud. Existing sampling techniques follow this same approach and allow us to bound the amount of error introduced for aggregation queries in each window [5].

Now, suppose that the data streams generated by D_1 and D_2, represented by random variables X_1 and X_2, exhibit strong linear correlation over time. This is expected, for example, if the devices are in close proximity and capturing temperature data. We can leverage this dependency by generating a compact representation of $\mathbb{E}[X_1|X_2]$ (or X_2 given X_1) at the edge and instruct the cloud to impute values based on this model. If the correlation is sufficiently strong, the edge may choose to exclusively forward samples from D_2 along with a compact representation of $\mathbb{E}[X_1|X_2]$. The cloud can then use the model to generate samples from D_1 without incurring any cost for transferring them (see Figure 2). This technique has the potential to reduce the network traffic required to obtain error rates comparable to other sampling mechanisms, provided we can determine how many samples to send and impute each window.

B. Optimization Problem

Our optimization framework allocates real samples to device streams (to forward to the cloud) and computes the number of samples to impute for each device to minimize the error introduced for aggregate queries. Table I outlines the notation used throughout this section with respect to a single tumbling window. We assume that the streams produce real-valued data; however, we make no strong assumptions about the data distribution of each device. We also assume that samples drawn from each device are independent and identically distributed within a single stream window (we relax this assumption in section IV). Then for each stream i, we seek the optimal number of real and imputed samples ($n_{r,i}$ and $n_{s,i}$), in addition to identifying a highly correlated stream to use as a predictor (p_i). We can then build a model for the expected value of X_i given X_{p_i}, which allows us to impute data from stream i given observations from stream p_i. We also include an upper bound on the number of values we should impute, given the quality of the model. Our full optimization problem is outlined in equation 1. We motivate and provide intuition for the objective and constraints in the following sections.

TABLE I: Notation

k	Number of streams available for sampling
X_i	A random variable following the probability distribution from stream i
N_i	The number of tuples that arrive from stream i
$n_{r,i}$	Number of real samples allocated to stream i
$n_{s,i}$	Number of imputed samples allocated to stream i
p_i	An index associated with stream i corresponding to a predictor stream which correlates with stream i
μ_i	The mean of stream i
σ_i^2	The variance of stream i
w_i	Weights identifying a stream's importance
$c_i(n,m)$	Cost to forward n samples and impute m samples for stream i (must be a *convex* function of n,m)
C	Upper bound on the traffic cost
ϵ_i	Bound on the reduction in variance introduced by imputation for stream i

$$\min_{n_r,\, n_s,\, p} \quad \sum_{i=1}^{k} w_i^2 \operatorname{Var}[\hat{\mu}_i] \tag{1a}$$

$$\text{s.t.} \quad p_i \in \{1, 2, ..., k\} - \{i\} \tag{1b}$$

$$0 \leq \; n_{r,i} \; \leq \; N_i \tag{1c}$$

$$0 \leq \; n_{s,i} \; \leq \; n_{p,i} \tag{1d}$$

$$(n_{r,i} + n_{s,i}) \; > 1 \tag{1e}$$

$$\sum_{i=1}^{k} c_i(n_{r,i},\, n_{s,i}) \; \leq \; C \tag{1f}$$

$$|\; \text{Bias}(\hat{\sigma}_i^2) \;| \; \leq \; \epsilon_i \tag{1g}$$

1) Objective Function: We seek to minimize error when estimating aggregate queries, subject to a bound on the traffic cost. Our problem statement requires us to minimize the error associated with many aggregate functions. However, the sampling distributions for certain aggregates (e.g. order statistics) depend on the data distribution itself, which we do not assume is known. Our proposal is to directly minimize the error associated with an AVG query while simultaneously controlling for the amount of bias introduced in our estimates for other aggregates. Minimizing the error for the mean necessarily reduces the error for many other aggregates. Errors for order statistics and percentiles are inversely proportional to the number of samples, so *increasing the effective sample size through accurate imputation necessarily reduces errors for these queries* [27]. Therefore, minimizing a weighted[3] sum of the squared errors for an AVG query across all the streams results in the objective:

$$f(n) = \sum_{i=1}^{k} w_i^2 \,\mathrm{Var}[\hat{\mu}_i] = \sum_{i=1}^{k} \frac{w_i^2 \sigma_i^2}{n_{r,i} + n_{s,i}} \tag{2}$$

Note that this formulation does not require us to make any strong assumptions about the distribution of each stream.

2) Constraints: For each stream i, the variable p_i is assigned the index of a different stream, which will be used as a candidate for prediction (constraint 1b). We require that each stream have an associated predictor stream, i.e. no predictor can be null. If the predictor stream correlates poorly with the target stream, constraint 1g will prevent us from performing imputation. This formulation only allows for a single stream to be used as a predictor; we discuss the possibility of using multiple predictors in section V-G.

Constraints 1c and 1d provide upper bounds on the number of real and imputed samples. We can't send more real samples than we observed at the edge and we can't impute more than the number of real samples taken from the predictor stream.

Constraint 1e ensures that we have at least one sample from each stream, regardless of if it is real or imputed. This is required for constraint 1g to be defined (see equation 5).

Constraint 1f requires that the number of real samples and compact models forwarded across all streams stays within our bound on WAN traffic. This constraint could be swapped with the objective function to produce a related optimization problem where we directly minimize network traffic subject to a constraint on the accuracy.

Our final constraint (1g) simultaneously addresses two related modeling concerns:

1) *Underestimating the variance*. Our models estimate the expected value of a target stream given a predictor stream. Therefore, each time we impute a value we are implicitly reducing the variability of the target stream sample and biasing the result of a VAR or MAX query. If the streams are not sufficiently correlated, we could severely underestimate the variability in the target stream.
2) *Model quality*. If we construct a poor model for the expected value, our imputed values will not be representative of the target stream. Therefore, we require a mechanism for controlling model quality.

To address both concerns, consider the following variance decomposition formula:

$$\mathrm{Var}[X_i] = \mathbb{E}[\mathrm{Var}[X_i|X_{p_i}]] + \mathrm{Var}[\mathbb{E}[X_i|X_{p_i}]] \tag{3}$$

[3] In our implementation, we set the weights to be inversely proportional to the expected value of each stream, thereby allowing us to minimize the coefficient of variation. This is a common approach which prevents us from disproportionately assigning samples to streams with high variance if the variance relative to the mean is low [28].

The $\mathrm{Var}[\mathbb{E}[X_i|X_{p_i}]]$ term captures the variance in X_i that is explained by our model for the expected value. Since $\mathrm{Var}[\mathbb{E}[X_i|X_{p_i}]] \leq \sigma_i^2$, we introduce a constraint that allows the user to dictate how large of a reduction in variance they can tolerate. This is achieved by computing an expression for the bias introduced by our imputation and bounding it above by a constant (ϵ_i). The expected reduction in variance associated with imputing $n_{s,i}$ values rather than sending them is equal to the bias of our variance estimator ($\hat{\sigma}_i^2$), given by:

$$\mathrm{Bias}(\hat{\sigma}_i^2) = \mathbb{E}\left[\frac{(n_{r,i}-1)\hat{\sigma}_{r,i}^2 + (n_{s,i}-1)\hat{\sigma}_{s,i}^2}{n_{r,i}+n_{s,i}-1}\right] - \sigma_i^2 \tag{4}$$

$$= \frac{(n_{s,i}-1)\,\mathrm{Var}[\mathbb{E}[X_i|X_{p_i}]] - n_{s,i}\sigma_i^2}{n_{r,i}+n_{s,i}-1} \tag{5}$$

Using this formula, we can readily compute how much we expect the variance to decrease given $n_{r,i}$ and $n_{s,i}$. In practice, the bias will heavily depend on the type of imputation being performed. If we simply replace missing values with a point estimate for the mean, then $\mathrm{Var}[\mathbb{E}[X_i|X_{p_i}]]$ is exactly zero. Model-based imputation techniques will almost always explain a positive amount of the variance.

3) Optimization at the Edge: Our optimization problem has a non-linear objective function and integer-valued variables, making it a non-linear integer program, which is computationally NP-hard to solve in general. Our system solves this problem at the edge after each aggregation window; therefore, computational efficiency is important. To simplify the problem, we propose treating p as a constant, i.e., we treat the optimal selection of stream predictors as a sub-problem, which we solve prior to solving the optimization problem. We propose a heuristic in section IV-A for obtaining a solution to this sub-problem efficiently, which leads to the following result.

Theorem. *The problem in equation 1 is a convex optimization problem when p is treated as a constant and the integer constraints are relaxed on the objective variables.*

Proof. Concatenate n_r and n_s to form a single vector $n = (n_r, n_s) \in \mathbb{R}^{2k}_{\geq 0}$. To simplify notation for the Hessian, for $i \in 1 \ \ldots \ k$, we define the non-negative variables $\psi_i = 2w_i^2\sigma_i^2/(n_i + n_{i+k})^3$. Then the non-zero Hessian entries are:

$$\frac{\partial^2 f}{\partial n_j \partial n_i} = \begin{cases} \psi_i & 1 \leq i \leq k \text{ and } j = i \\ \psi_{i-k} & k \leq i \leq 2k \text{ and } j = i \\ \psi_i & 1 \leq i \leq k \text{ and } j = i+k \\ \psi_{i-k} & k \leq i \leq 2k \text{ and } j = i-k \end{cases}$$

For arbitrary $z \in \mathbb{R}^{2k}$, we have:

$$
\begin{aligned}
z^T(\nabla^2 f(n))z &= \sum_{i=1}^{k}\sum_{j=1}^{k} z_i z_j (\nabla^2 f(n))_{ij} \\
&= \sum_{i=1}^{k} \psi_i (z_i^2 + z_{i+k}^2) + \sum_{i=1}^{k} 2\psi_i z_i z_{i+k} \\
&= \sum_{i=1}^{k} \psi_i (z_i + z_{i+k})^2 \ \geq 0
\end{aligned}
$$

Therefore, the Hessian is positive semi-definite and the objective function is convex. Given the properties of our constraints and the fact that $\mathbb{R}^{2k}_{\geq 0}$ is a convex set, the problem is a convex optimization problem [29].

This convexity result gives us some assurance that the optimization problem can be solved efficiently, which is important given the heterogeneous compute resources available at the edge. Our sampling algorithm at the edge proceeds according to the flow outlined in Algorithm 1.

Algorithm 1: Stream Sampling Algorithm

Input : Window Duration, t; Cost bound, C, Variation Reduction Bounds, ϵ_i
Output: Samples and models to send to the cloud
Start Timer for t seconds
while *Aggregation timer is running* **do**
  Insert inbound samples into cache
end
Estimate σ_i^2
Use heuristic to select predictors for each stream
Solve optimization problem in eq. 1 for $n_{r,i}$ and $n_{s,i}$
Forward samples and compact models to cloud

IV. Practical Heuristics

There are some barriers to directly applying our optimization framework in real-world environments. We now discuss some of those issues and how to handle them in practice.

A. Predictor Selection

Our optimization problem can be solved efficiently if we treat the predictor indexes as constants. If we wanted a globally optimal solution, it would require considering $O(k!)$ possible permutations for the predictors. We use a polynomial-time heuristic that can be used to obtain a set of predictors more efficiently. For a given stream X_i, we simply select, as a predictor, the stream which correlates the highest with X_i, which requires $O(k^2)$ comparisons. This does not necessarily yield the globally optimal solution; it may be better to choose a slightly sub-optimal predictor for a given stream to reduce the total number of overall predictors, which allows us to send real samples for a small number of predictor streams. We evaluate the accuracy of this heuristic and its impact on latency in sections V-B and V-E.

B. Modeling and Dependence Estimation

There are many different kinds of predictive models that could be used to impute values from a stream. We assume that the model estimates the expected value $\mathbb{E}[X_i|X_{p_i}]$ and can be represented *compactly*. An appropriate model will also depend on the measure of dependence used by the implementation:

1) If we use the *Pearson* correlation coefficient as our measure of dependence, a simple linear model will suffice for estimating the expected value.
2) If we use the *Spearman* correlation as our measure of dependence, a natural model choice could be polynomial regression. Modeling the response stream based on a 3rd degree polynomial provides compactness and the flexibility to fit a variety of monotonic functions.

Our framework allows for arbitrarily complex models, but our evaluation shows empirically that these simple models can provide substantial benefit in practice. Furthermore, increasing model complexity will increase traffic over the WAN to send model parameters, require more computation at the edge for dependence estimation and model fitting, and increase model inscrutability. Therefore, *simple models are preferable in our proposed system.*

C. Bounding the Bias

The ϵ_i constants form upper bounds on the amount of bias we are willing to tolerate in our variance estimates for each stream. While our estimator for the variance is necessarily biased after imputation, that does not imply that, on average, it deviates substantially from the true variance. One method for computing ϵ_i would be to use a percentage of the observed variance in the stream. For example, we could specify $\alpha = 0.05$ and set $\epsilon_i = \alpha\sigma_i^2$ which allows us to bias the variance estimate up to 5% of the observed variance. To avoid the problem of selecting α for each application, we consider a heuristic based on how much uncertainty is in our estimate of σ_i^2 at the edge. In practice, we actually compute two estimators for the variance in our framework: one at the edge and one in the cloud. Since this edge estimate is necessarily an imperfect representation of what was observed on the device, we can compute the variance of the edge estimator as follows [30]:

$$
\text{Var}[\hat{\sigma}_i^2] = \frac{1}{N_i}\left(\mu_4 - \frac{N_i - 3}{N_i - 1}\sigma_i^4\right) \tag{6}
$$

where μ_4 is the fourth central moment of X_i. We consider selecting $\epsilon_i = \sqrt{\text{Var}[\hat{\sigma}_i^2]}$, which requires the bias to fall within one standard error of the unbiased estimate. The intuition is that we allow the amount of bias in the cloud estimator to scale with the amount of uncertainty in our edge estimator; if we have a precise estimate of the variance at the edge, we will force the optimization framework to avoid substantially biasing the estimate computed in the cloud. Similarly, if our estimate at the edge is noisy, we will allow more bias to be introduced in the downstream estimate. We explore the implications of different selection strategies in our evaluation.

D. Independence Assumption

Our framework assumes the independence of samples obtained from each stream in a given window. We believe this assumption is sometimes justified in practice, since measurement noise may dominate changes in the signal during a sufficiently small time interval. However, this is an unreasonable expectation for most time-series applications. In practice, a *thinning* technique [31] could be used to address this dependence, which is a common method used for obtaining independent samples from a Markov chain.

We could also weaken the independence assumption to allow for m-dependence, where data points that are more than m lags apart are assumed to be independent [32]. This allows us to add an error term to our objective function which is proportional to the strength of the dependence:

$$\text{Var}[\hat{\mu}_i] = \sigma_i^2 + 2\sum_{j=1}^{m} \text{Cov}[x_{i,1}, x_{i,1+j}] \tag{7}$$

The number of additional terms in this sum is linear in m (and a constant with respect to our optimization objective). Therefore, the convexity of our problem is unaffected.

V. Evaluation

A. Experimental Setup

1) Testbed and Methodology: We simulate an edge device using a local machine equipped with an Intel i7-7500U processor and constrained to have 1 GB of RAM. Inbound streams are constructed by reading data from files in fixed size windows. At the end of each window, we call into an optimization module which solves for the real and imputed sample counts. To test the end-to-end system, we stream this data over the WAN using Amazon Kinesis [33]. Our t2-medium EC2 instance in the cloud runs Apache Storm [34], which consumes and processes data from Kinesis.

2) Datasets: We consider three real-world datasets for our evaluation: the *Home dataset*, the *Turbine dataset*, and the *Smart City dataset*. The Home dataset contains temperature measurements taken from three homes in Massachusetts [35]. The Turbine dataset consists of sensor traces collected from wind turbines in France [36]. These sensors collect various measurements, including temperatures, power, wind speed, rotor speed, among other things. The Smart City dataset contains measurements across a diverse set of devices located in Aarhus, Denmark [16]. We aggregated data from several different devices, including:

- Weather sensors, which measure a variety of quantities including temperature and humidity.
- Pollution sensors, which capture the amount of a chemical compound in the air.
- Parking lot sensors, which measure the current capacity of a specific lot.
- Traffic sensors, which report counts of cars that travel past a specific location.

We prefer the smart city dataset for examining sensitivity, since it is noisy and representative of a real-world scenario.

3) System Comparisons: We consider the following streaming systems in our evaluation:

- *ApproxIoT:* A stream-sampling system which performs a variant of stratified sampling at the edge [5]
- *S-VOILA:* A stream-sampling system which allocates sample sizes based on the observed variability in each stream [23]
- *Mean:* Our proposed system with edge sampling and cloud-side imputation using the mean
- *Model:* Our proposed system with edge sampling and cloud-side imputation using a compact model

For each sampling technique, we systematically vary the amount of data we can send to the cloud and report the error rates for each data size. Unless otherwise specified, our proposed method defaults to using the Spearman correlation as our dependence measure and a cubic representation of the conditional expectation. By default, we set the ϵ_i bounds equal to one standard error of the edge variance estimate, as discussed in section IV-C.

4) Queries and Metrics: We use a variety of aggregate queries to evaluate our framework, including AVG, VAR, MIN, and MAX. As an example, a query for the average value in a window for each device can be written as:

```
SELECT AVG(response), dev_id, win_id
FROM Response_Table
WHERE win_id = current_window
GROUP BY dev_id;
```

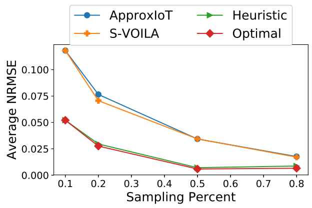
If we have temperature data, these queries could be used to obtain the average or max temperature on a given day. In a smart city setting, we might be interested in quantifying the variability associated with traffic patterns. The MIN and MAX queries are important for applications that are interested in outlier detection or top-k results. We deliberately exclude COUNT queries, since our framework can be easily modified to behave similarly to the ApproxIoT system, which can provide exact answers [5]. We emphasize that our framework supports other aggregates and arbitrary selection predicates, since we are performing sampling. We chose the AVG, VAR, and MIN/MAX queries to evaluate our framework's ability to capture the average behavior and the variability present in each stream (including extreme outliers). To quantify the error for each query, we compute the *normalized root mean square error (NRMSE)* , given by:

$$\text{NRMSE}_i = \frac{\sqrt{\frac{1}{T}\sum_{j=1}^{T}(\hat{\theta}_{ij} - \theta_{ij})^2}}{\bar{\theta}_i} \tag{8}$$

where θ_{ij} is the true value of the aggregate for device i in window j and T is the total number of windows in the experiment. We require a normalized RMSE, since each device may have a radically different expected value and variance.

B. Heuristic vs. Optimal Predictor Selection

We first evaluate our predictor selection heuristic. Since there are a factorial number of predictor combinations, we use



(a) Error rates for an AVG query.

Method	Average NRMSE	Gain
ApproxIoT	0.077	0.0%
S-VOILA	0.071	7.8%
Heuristic	0.030	61.0%
Optimal	0.028	63.6%

(b) Performance with a 20% sampling rate.

Fig. 3: Heuristic performance on the Home dataset.

only three highly correlated streams from the Home dataset to measure how well our heuristic approximates the global solution.

Figure 3a compares AVG query errors across different sampling rates. We observe a slight drop in accuracy when using the heuristic compared to the global solution. Figure 3b shows the exact difference in errors for the 20% sampling rate. The maximum difference between the heuristic and optimal error reductions across all sampling rates was 3.5%. Our heuristic performs comparably on the other two datasets, with a maximum degradation around 4%.

C. Turbine Dataset

The Turbine dataset contains measurements from a diverse set of sensors. This application is a perfect fit for our framework, given that bandwidth is often congested in rural locations and low latency is required to react to failures promptly [15]. Since these sensors are in the same location, we readily observe correlation in their measurements and expect our framework to leverage it. The strength of the pairwise linear correlations vary substantially, with some pairs less than 0.05, several pairs in the 0.3-0.5 range, and a few around 0.9.

Figure 4 shows the performance of our strategies across various sample allowances and aggregate queries. For the AVG query in figure 4a, we observe that if the user can tolerate an NRMSE of 0.1, our model imputation approach can obtain that level of accuracy sending 16% of the data, while ApproxIoT requires 32% of the data, which represents a 50% decrease. However, this performance varies based on the application's error bound, e.g. obtaining an NRMSE of 0.15 and 0.05 requires 44% and 60% less data respectively, when compared to ApproxIoT approach. We observe that S-VOILA consistently outperforms ApproxIoT, since it considers the variance in each stream. Both of our methods also outperform the S-VOILA system, with the model method requiring 27% and 36% less data to obtain an NRMSE of 0.1 and 0.05 respectively.

Figures 4b, 4c, and 4d explore how our minimization of the average impacts queries that depend on an accurate representation of the variability. In all cases, we observe an accuracy drop with mean imputation, especially for the VAR query. This is expected, given that imputing values with a constant will necessarily have a greater impact on the variability. The S-VOILA technique is comparable to our model method when at least 80% of the data is sent over the network. However, on smaller sample sizes our technique outperforms the baseline sampling methods on all four aggregate queries. This effect is caused by the relative importance of the imputed samples. When data is scarce, each imputed sample provides substantial information about the data distribution. This information becomes less important as the number of real samples increases.

D. Smart City Dataset

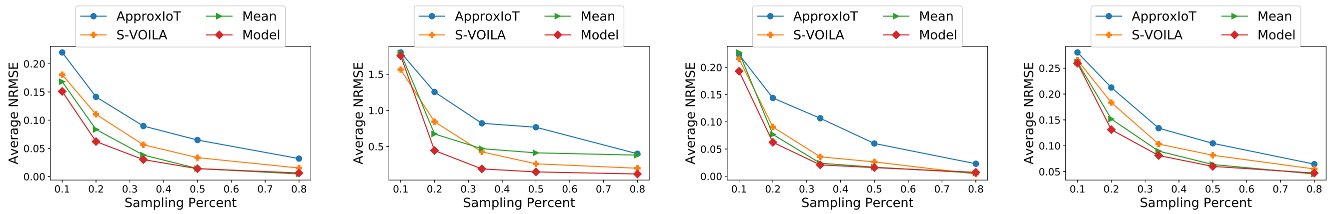
For this experiment, we consider devices which have radically different data distributions. We observed modest correlations between devices that measure different quantities, e.g., between parking lot occupancy and temperature (0.4 - 0.6). The strength of these correlations vary across stream windows.

Figure 5 shows the performance of our strategies across various sample allowances and aggregate queries. For the AVG query in figure 5a, we observe that if the user can tolerate an NRMSE of 0.1, our model imputation approach obtains that level of accuracy sending 18% of the data, while ApproxIoT requires 26% of the data, which represents a 30% decrease. This performance varies based on the application's error bound, e.g. obtaining a NRMSE of 0.05 requires 42% less data compared to ApproxIoT. Both of our methods also outperform the S-VOILA system, with the model method requiring 18% less data to obtain an NRMSE of 0.1 and 42% less data for an NRMSE of 0.05. We also note that the mean imputation technique begins to outperform the modeling technique once the sample size is sufficiently large. This is expected, since higher sample sizes will allow us to perform more imputation without completely removing the variability from the data. If the number of imputed samples allowed for the model and mean methods are the same, the mean imputation method will necessarily produce a smaller AVG error while inflating the error for the variance. This effect can be observed in figures 5b, 5c, and 5d, where model imputation outperforms the mean approach. We observe that our model imputation approach outperforms the baseline sampling methods on all four aggregate queries.

E. Computational Overhead

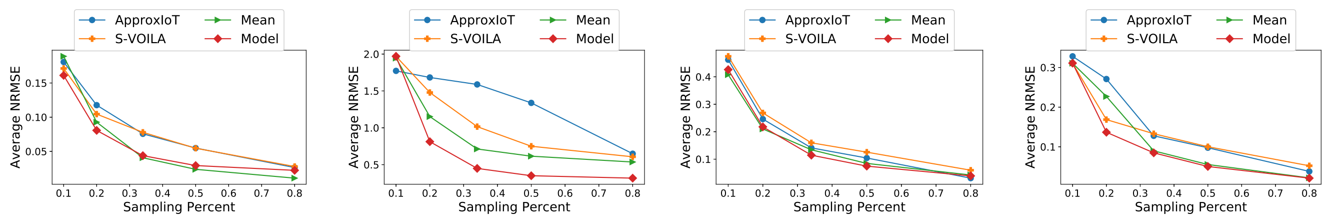
Our system introduces an overhead after each stream window to model dependence and compute optimal sample sizes at the edge. There is also a computational cost incurred on the cloud side to perform imputation; however, our system uses compact models which are cheap to evaluate. Clouds also have substantial computational resources available to the user, so we exclude it from this analysis. We used the sequential least squares programming (SLSQP) solver in the Python scipy library to perform the optimization [37].

Figure 6a shows how the overall latency scales with an increasing number of streams. We fixed the arrival frequency at 12 points per stream (e.g. one sample every 5 seconds with a window length of one minute). We observe that the overall latency is less than 400 milliseconds for both imputation methods with a stream count of 50. Solving the optimization problem accounts for the vast majority of the



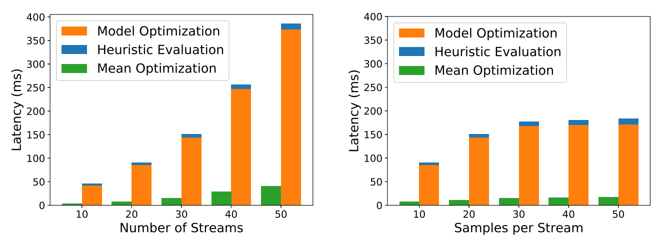
(a) Error for an AVG query (b) Error for a VAR query (c) Error for a MIN query (d) Error for a MAX query

Fig. 4: Turbine Dataset



(a) Error for an AVG query (b) Error for a VAR query (c) Error for a MIN query (d) Error for a MAX query

Fig. 5: Smart City Dataset



(a) Number of streams vs latency. (b) Samples per stream vs latency.

Fig. 6: System Latency

latency compared to the time required to model dependence with our proposed heuristic. We also note that mean imputation requires substantially less time to optimize compared to our model imputation method. There are two reasons for this effect: (1) mean imputation introduces more bias, so we are allowed to impute fewer data points, which restricts the search space in the optimization and (2) almost no time is required to evaluate the heuristic, since the degree of correlation between streams does not impact imputation accuracy. These latencies are unlikely to be an issue if the number of devices is relatively low or the window duration is sufficiently high. Applications that are latency sensitive may need to consider additional parallelism or ensure each edge has a manageable number of client devices.

Figure 6b shows the effect of arrival frequency on latency. We observe some effect; however, its impact on performance doesn't appear to grow without bound. We conclude that the dimension of the optimization variable (i.e. the number of streams) has the biggest impact on latency.

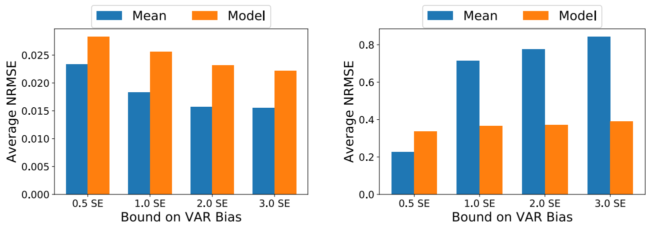
F. *Sensitivity Analysis*

Tolerance for Variance Bias. Modeling the expected value biases the variance estimator by reducing the variability present in the sample. We now examine how different tolerances for bias impact error rates using the Smart City dataset and a sampling rate of 50%. The tolerance values are expressed as a multiple of the standard error (SE) observed in the edge variance estimator, as discussed in section IV-C.

Figure 7a summarizes the error for an AVG query for our mean and model imputation techniques. For both methods, we observe smaller AVG errors as our tolerance for bias increases. This is expected, since a higher tolerance implies we are allowed to perform more imputation with our model for the expected value.

Figure 7b shows the impact of these bounds on a VAR query. We observe a steady increase in the error as our ability to bias the result increases. In addition, model imputation method has a smaller impact on the error, since it models the variability more accurately. We also note that mean imputation outperforms model imputation at the 0.5 SE level. This is caused by the difference in imputation frequency between the methods: model imputation is allowed to perform almost 3x as much imputation at that level, since it explains more of the variance. However, this introduces more bias in the variance query. Each application will have a different tolerance for biasing variance queries. Applications mostly interested in the average values will be able to allow higher amounts of imputation compared to those concerned with outliers.

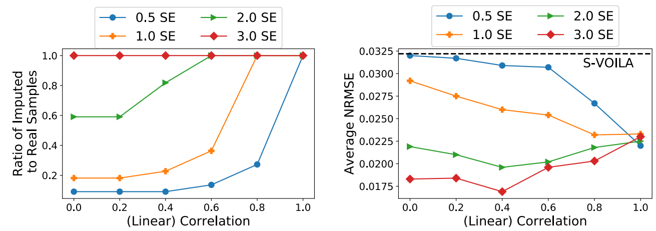
Correlation Effects. The efficacy of our approach depends on our ability to identify correlation in the data and exploit it. We examine the role of correlation in our framework by generating



(a) Error for a AVG query.

(b) Error for a VAR query.

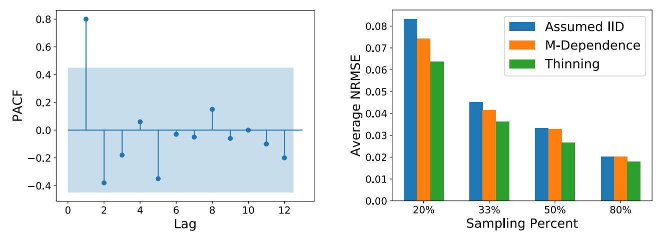
Fig. 7: Bounding the bias of the variance estimator.



(a) Imputation allowed.

(b) Errors for an AVG query.

Fig. 8: Correlation effects on imputation.



(a) PACF for the NO_2 stream.

(b) Errors for an AVG query.

Fig. 9: IID experiments Smart City Dataset

a synthetic dataset and varying the correlation present in the data. For these experiments, we use two streams drawn from a multivariate normal distribution with means equal to 30 and the diagonal elements of the covariance matrix equal to 16. We systematically vary the off-diagonal elements of the covariance matrix to obtain the desired correlation.

Figure 8a shows how much imputation we are allowed to perform across multiple values for bias tolerance and correlation. For 1 standard error, the amount of imputation allowed slowly increases as the strength of the correlation increases. When the correlation is above 0.8, we are allowed to impute a point for every real sample we send over the network. For 3 standard errors, there is no restriction on imputation, even if the correlation is zero. Using a small tolerance value of 0.5 results in more conservative behavior, requiring very strong dependence before significant imputation is performed.

Figure 8b shows the AVG query error rates across correlation levels. Our imputation methods always perform as well as the S-VOILA system, regardless of correlation strength. For smaller bias values (0.5 and 1.0 SE) we obtain a steady decrease in error as the correlation between streams increases. However, for larger bias values (2.0 and 3.0 SE), there is a more complex relationship between correlation and error. The largest errors for these values is obtained when the streams are very strongly correlated. This is explained by considering the difference between mean imputation and model imputation. When the correlation is very low, these techniques are essentially performing mean imputation, since only a small fraction of the variance is explained by the model. As the strength of the correlation increases, the model begins explaining more of the variance, but is a noisier representation of the expected value when compared to mean imputation.

IID Assumption. Time series data often exhibit autocorrelation; however, our framework assumed that samples for each device were IID within a stream window. In section IV-D we discussed options for relaxing this assumption in practice, including implementing a thinning mechanism or adding a penalty term for the autocorrelation. We use the Smart City dataset to evaluate the performance of these options in practice.

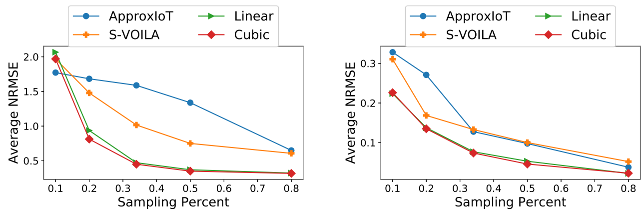
Figure 9a displays a partial autocorrelation function (PACF), which shows the strength of the autocorrelation for one of the pollution sensors. There is only one statistically significant lag with a positive correlation around 0.8, so we let $m = 1$ for this experiment when estimating m-dependence.

Figure 9b compares the performance of these techniques on an AVG query. The thinning technique consistently obtains the lowest error rate. Furthermore, thinning works without significant tuning from the user. Implementing m-dependence in practice would require dynamically computing a PACF to estimate the number of significant lags in a time series. We therefore recommend the thinning technique in practice.

Linear vs Cubic Models. In section IV-B, we discussed options for measuring dependence between device streams and *compact* representations of the conditional expectation. Our experiments used the cubic representation by default; however, we observe no significant difference between the two methods on an AVG query. The results of the VAR and MAX queries are show in figures 10a and 10b respectively. The cubic models provide a slight improvement (around 3%) when estimating these queries due to their ability to capture the tails of the distributions more accurately. A future enhancement could use LASSO or some other regularization method to dynamically trade-off between linear and cubic models [38].

G. Discussion

Our sampling and imputation technique generates much less WAN traffic and obtains comparable error rates compared to other stream sampling systems. The bias tolerance ϵ_i is an important tuning parameter, which bounds the amount of bias in the variance estimator and implicitly dictates the model quality required to perform imputation. The correlation strength also affects our ability to minimize WAN traffic; high correlations between streams provides more opportunity for cost savings.



(a) Errors for a VAR query (b) Errors for a MAX query

Fig. 10: Linear vs Cubic models with the Smart City Dataset

Our framework is restricted to using a single predictor stream when building models for imputation. It is conceivable that using multiple streams as predictors could produce better models and allow us to impute more values. In this work, we assume model estimation is performed locally at the edge. Leveraging multiple predictor streams increases the complexity of model fitting and increases the size of the model that must be sent over the WAN. In this case, our framework could potentially be modified to leverage historical data in the cloud to identify more complicated relationships between streams. However, our experiments provide some empirical evidence that using a single predictor stream can provide significant cost savings in practice.

VI. Related Work

A. Edge and Stream Sampling

Stream sampling is a common technique for reducing network traffic. Reservoir sampling is frequently used to obtain a variety of probability samples over unbounded streams [20], [39]. A similar work considers ways in which a central node can obtain a random sample from geo-distributed edge nodes and establishes lower bounds on the number of messages that must be sent [40], [41]. A wide variety of other works exist in the field of Approximate Query Processing which seek to systematically trade accuracy for performance [42], [43]. The ApproxIot and S-VOILA streaming systems both leverage more sophisticated sampling strategies to address properties of individual streams. We improve on these algorithms by supplementing the samples using imputations with explicit bounds on the error.

B. Sensor Networks

A related area of research considers methods for efficiently obtaining samples from a sensor network. Energy constraints are the main issue as opposed to bandwidth, but the objective is the same: minimize the query error subject to some cost function. The TinyDB system leverages the estimated energy cost and variation for a given device when executing queries [44]. Dependence is another relevant consideration for this problem, since sensors in the same sensor network will likely produce correlated observations. Systems similar to BBQ are designed with this dependence in mind and attempt to exploit it to reduce the energy requirement [25], [45]. We perform an analogous task over dependent data streams by leveraging correlation to reduce the amount of data required to answer queries. Futhermore, we do this in an online fashion (with no offline profiling) and do not assume a parametric model or anything about the underlying data distributions.

C. Sketch Summaries

Sketching is a widely used technique for computing compact, approximate representations of data [46], [47]. In some cases, approximation is not required; for example, moment sketches can exactly estimate the mean and variance in constant space and time [48]. However, our framework makes no assumptions about the downstream operations a user will perform on the resulting data. If a user wants to perform centralized machine learning in the cloud, our framework would provide a dataset which maintains dependencies between streams. Lower fidelity representations, including sketches, would not be as amenable to these use cases.

VII. Conclusion

Given the increase in data volume and practical WAN bandwidth constraints, efficient data transfer continues to be an important research direction. We present a system that leverages correlation between streams for supplementing samples with accurate imputations in the cloud to reduce WAN traffic. Our evaluation suggests it could reduce bandwidth costs by anywhere from 27% to 42%, depending on the application. While our implementation and evaluation focuses on real-valued data, we believe this general approach can be applied in other streaming settings, such as video streams. Camera deployments often exhibit temporal/spatial correlation and must contend with limited WAN resources [49]. Given these constraints and additional compute constraints at the edge, it may be preferable to use correlated video streams to obtain approximate query results rather than performing expensive inference at the edge or streaming frames to the cloud for analysis. We leave further exploration of this application to a future work. Computing and leveraging dependence in these settings could allow for an even broader set of applications to benefit from this approach.

Acknowledgement

This research was supported in part by the NSF under grants CNS-1717834 and CNS-1908566.

References

[1] Statista I.H.S., "Internet of things (iot) connected devices installed base worldwide from 2015 to 2025 (in billions)," 2018.

[2] D. Kumar, S. Ahmad, A. Chandra, and R. K. Sitaraman, "Aggnet: Cost-aware aggregation networks for geo-distributed streaming analytics," *ACM/IEEE Symposium on Edge Computing (SEC'21)*.

[3] S. H. Mortazavi, M. Salehe, M. Gabel, and E. d. Lara, "Feather: Hierarchical querying for the edge," in *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, 2020, pp. 271–284.

[4] P. Kang, P. Lama, and S. U. Khan, "Slo-aware virtual rebalancing for edge stream processing," in *2021 IEEE International Conference on Cloud Engineering (IC2E)*, 2021, pp. 126–135.

[5] Z. Wen, D. L. Quoc, P. Bhatotia, R. Chen, and M. Lee, "Approxiot: Approximate analytics for edge computing," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, 2018, pp. 411–421.
[6] P. Lou, L. Shi, X. Zhang, Z. Xiao, and J. Yan, "A data-driven adaptive sampling method based on edge computing," *Sensors*, vol. 20, no. 8, p. 2174, Apr 2020.
[7] D. Trihinas, G. Pallis, and M. D. Dikaiakos, "Adam: An adaptive monitoring framework for sampling and filtering on iot devices," in *2015 IEEE International Conference on Big Data (Big Data)*, 2015, pp. 717–726.
[8] S. Memon and M. Maheswaran, "Optimizing data transfers for bandwidth usage and end-to-end latency between fogs and cloud," in *2019 IEEE International Conference on Fog Computing (ICFC)*, 2019, pp. 107–114.
[9] L. Mesin, S. Aram, and E. Pasero, "A neural data-driven algorithm for smart sampling in wireless sensor networks," in *EURASIP Journal on Wireless Communications and Networking*, 2014.
[10] J. Hribar and L. DaSilva, "Utilising correlated information to improve the sustainability of internet of things devices," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, 2019, pp. 805–808.
[11] M. H. Mazhar and Z. Shafiq, "Characterizing smart home iot traffic in the wild," 2020.
[12] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-IID data quagmire of decentralized machine learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 4387–4398.
[13] P. Fraga-Lamas, T. M. Fernndez-Carams, and L. Castedo, "Towards the internet of smart trains: A review on industrial iot-connected railways," *Sensors*, vol. 17, no. 6, 2017.
[14] S. Tata, R. Jain, H. Ludwig, and S. Gopisetty, "Living in the cloud or on the edge: Opportunities and challenges of iot application architecture," in *2017 IEEE International Conference on Services Computing (SCC)*, 2017, pp. 220–224.
[15] M. Schlechtingen, I. F. Santos, and S. Achiche, "Using data-mining approaches for wind turbine power curve monitoring: A comparative study," *IEEE Transactions on Sustainable Energy*, vol. 4, no. 3, pp. 671–679, 2013.
[16] S. Kolozali *et al.*, "A knowledge-based approach for real-time iot data stream annotation and processing," September 2014.
[17] M. K. Geldenhuys, J. Will, B. J. J. Pfister, M. Haug, A. Scharmann, and L. Thamsen, "Dependable iot data stream processing for monitoring and control of urban infrastructures," in *2021 IEEE International Conference on Cloud Engineering (IC2E)*, 2021, pp. 244–250.
[18] K. Patroumpas and T. Sellis, "Window specification over data streams," in *Proceedings of the 2006 International Conference on Current Trends in Database Technology*, ser. EDBT'06. Berlin, Heidelberg: Springer-Verlag, 2006, p. 445464.
[19] D. Kumar, J. Li, A. Chandra, and R. Sitaraman, "A ttl-based approach for data aggregation in geo-distributed streaming analytics," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 2, Jun. 2019.
[20] M. Al-Kateb and B. S. Lee, "Stratified reservoir sampling over heterogeneous data streams," in *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management*, ser. SSDBM'10. Berlin, Heidelberg: Springer-Verlag, 2010, p. 621639.
[21] M. Al-Kateb and B. Lee, "Adaptive stratified reservoir sampling over heterogeneous data streams," *Inf. Syst.*, vol. 39, p. 199216, Jan. 2014.
[22] D. L. Quoc *et al.*, "Approxjoin: Approximate distributed joins," in *Proceedings of the ACM Symposium on Cloud Computing*, ser. SoCC '18. New York, NY, USA: ACM, 2018, pp. 426–438.
[23] T. Nguyen *et al.*, "Stratified random sampling over streaming and stored data," *Advances in Database Technology - 22nd International Conference on Extending Database Technology (EDBT)*, Mar 2019.
[24] M. Bermudez-Edo, P. Barnaghi, and K. Moessner, "Analysing real world data streams with spatio-temporal correlations: Entropy vs. pearson correlation," *Automation in Construction*, vol. 88, pp. 87–100, 2018.
[25] A. Deshpande *et al.*, "Model-driven data acquisition in sensor networks," in *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, ser. VLDB '04, 2004, p. 588599.
[26] L. Cohen *et al.*, "Real-time data mining of non-stationary data streams from sensor networks," *Information Fusion*, vol. 9, no. 3, pp. 344–353, 2008, special Issue on Distributed Sensor Networks.
[27] H. A. David and H. N. Nagaraja, *Order Statistics, Third Edition*. Wiley, 2003.
[28] T. D. Nguyen, M.-H. Shih, S. S. Parvathaneni, B. Xu, D. Srivastava, and S. Tirthapura, "Random sampling for group-by queries," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, 2020, pp. 541–552.
[29] S. Boyd and L. Vandenberghe, *Convex Optimization*. USA: Cambridge University Press, 2004.
[30] A. Mood *et al.*, *Introduction to the Theory of Statistics*, ser. International Student edition. McGraw-Hill, 1973.
[31] A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin, *Bayesian Data Analysis, Third Edition*, ser. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis.
[32] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications (Springer Texts in Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2005.
[33] "Amazon kinesis." [Online]. Available: https://aws.amazon.com/kinesis/
[34] "Apache storm." [Online]. Available: https://storm.apache.org/
[35] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht, "Smart*: An open data set and tools for enabling research in sustainable homes," ser. ACM Proc. SustKDD, 2012.
[36] E. Group, "La haute borne data," 2020, data retrieved from ENGIE, https://opendata-renewables.engie.com/.
[37] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online]. Available: http://www.scipy.org/
[38] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996. [Online]. Available: http://www.jstor.org/stable/2346178
[39] J. S. Vitter, "Random sampling with a reservoir," *ACM Trans. Math. Softw.*, vol. 11, no. 1, p. 3757, Mar. 1985.
[40] S. Tirthapura and D. P. Woodruff, "Optimal random sampling from distributed streams revisited," in *Distributed Computing*, D. Peleg, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 283–297.
[41] R. Jayaram *et al.*, "Weighted reservoir sampling from distributed streams," in *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, ser. PODS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 218235.
[42] S. Agarwal *et al.*, "Blinkdb: Queries with bounded errors and bounded response times on very large data," in *Proceedings of the 8th ACM European Conference on Computer Systems*, ser. EuroSys '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 2942.
[43] S. Acharya, P. B. Gibbons, and V. Poosala, "Congressional samples for approximate answering of group-by queries," *SIGMOD Rec.*, vol. 29, no. 2, p. 487498, May 2000.
[44] S. R. Madden *et al.*, "Tinydb: An acquisitional query processing system for sensor networks," *ACM Trans. Database Syst.*, vol. 30, no. 1, p. 122173, Mar. 2005.
[45] L. A. Villas *et al.*, "A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks," *Ad Hoc Networks*, vol. 12, pp. 69–85, 2014.
[46] G. Cormode and M. Garofalakis, "Sketching streams through the net: Distributed approximate query tracking," in *Proceedings of the 31st International Conference on Very Large Data Bases*, ser. VLDB '05. VLDB Endowment, 2005, p. 1324.
[47] A. Dobra *et al.*, "Processing complex aggregate queries over data streams," in *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 6172.
[48] E. Gan, J. Ding, K. S. Tai, V. Sharan, and P. Bailis, "Moment-based quantile sketches for efficient high cardinality aggregation queries," *Proc. VLDB Endow.*, vol. 11, no. 11, p. 16471660, Jul. 2018.
[49] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: Scalable adaptation of video analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 253266.