# Approximate Fault Tolerance for Edge Stream Processing

Daiki Takao(✉), Kento Sugiura, and Yoshiharu Ishikawa

Graduate School of Informatics, Nagoya University, Aichi, Japan
takao@db.is.i.nagoya-u.ac.jp, {sugiura,ishikawa}@i.nagoya-u.ac.jp

**Abstract.** Existing distributed stream processing systems generally guarantee fault tolerance by switching to standby machines and reprocessing lost data. In edge computing environments, however, we have to duplicate each edge for this conventional approach. This duplication cost increases sharply with expansion in the system scale. To solve this problem, we propose an approach to support *approximate fault tolerance* without edge duplication. We focus on environmental monitoring applications and utilize the correlation between sensors. In this paper, we assume that each edge estimates missing data from the observed data and aggregates them approximately. We provide a method to estimate the outputs of failed edges taking care of the uncertainty of the processing results at each edge. Our method allows the server to continue processing without waiting for the recovery of failed edges. We also show that the validity of our method by experiments using synthetic data.

**Keywords:** Data stream processing · Edge computing · Fault tolerance

## 1 Introduction

Fault tolerance is an essential requirement for distributed data stream processing systems. Existing systems, such as Flink [2], process dynamically generated data in real-time by pipeline processing. Each task distributed to multiple servers receives inputs from the previous one and sends the processing results to the next one. This processing model enables efficient data processing, but a failure in one task affects the entire system. Since many applications, such as environmental monitoring, require high availability and low latency, processing systems need a mechanism to continue processing even if each server fails.

Existing systems generally guarantee fault tolerance by switching to standby nodes and reprocessing lost data [2,3,6,12]. When the manager detects a worker failure, a newly launched worker takes over the processing of the failed one.

This paper is based on results obtained from a project, JPNP16007, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). Also, This work was partly supported by KAKENHI (16H01722 and 20K19804).


G. Kotsis et al. (Eds.): DEXA 2021 Workshops, CCIS 1479, pp. 173–183, 2021.
https://doi.org/10.1007/978-3-030-87101-7_17

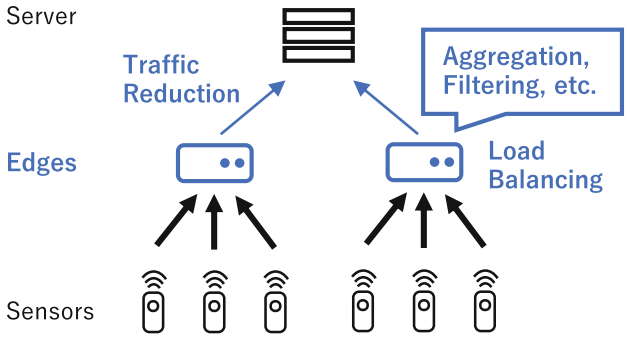


**Fig. 1.** Overview of the edge computing

Besides, some systems can recover from the failure without error by reprocessing only the lost data. This approach is suitable for cluster environments where the system can manage all resources centrally and assign the input data management task to the message queue.

In edge computing environments, however, we need a different approach to guarantee fault tolerance. Edge computing is a new processing model for traffic reduction and load balancing [15]. Each edge performs simple data processing such as aggregation and filtering, as shown in Fig. 1. For example, consider environmental sensing applications in a smart city. Environmental sensors widely installed in various places send measurements to the nearby edge by radio. Edges aggregate the inputs at regular intervals and send the result to the server. When an edge fails, another one at a distance cannot receive the measurements and therefore cannot take over the processing instead. That is, we have to duplicate each edge for the conventional approach: switching to standby nodes and reprocessing lost data. This duplication cost increases sharply with expansion in the system scale, unlike the cloud environment, which can manage all resources centrally.

To solve this problem, we focus on environmental monitoring applications and propose an approach to guarantee *approximate fault tolerance* without edge duplication. We deal with various failures by estimating missing data using the *correlation* between sensors rather than reprocessing. This approach guarantees the error bounds for a user-specified confidence threshold and allows the server to continue subsequent processing without waiting for the inputs from failed edges.

In this paper, we focus on aggregation queries and assume that some measurements are lost due to sensor/communication failures. Edges can handle these failures by estimating missing data from observed data and aggregating them approximately [8]. For example, if sensor D fails in Fig. 2, edge 2 estimates missing data from the measurements of sensor C and continues aggregation without waiting for the inputs from sensor D.

We extend this approach to the problem of edge failures. That is, the server estimates the outputs of failed edges from the others to continue processing. In Fig. 2, aggregation results for sensors A and B are lost due to the failure of edge 1. The server estimates them from the results for sensors C and D obtained from edge 2 and continues the subsequent processing without waiting for the recovery of edge 1.

In our approach, due to the approximate aggregation at each edge, the server has to take care of the uncertainty of the aggregation results to predict the error bounds accurately. There are various techniques to estimate missing values, such as mean imputation [10] and Gaussian process regression [14]. However, these techniques cannot take care of the uncertainty of the estimation in each edge, which causes the lack of reliability. On the other hand, our method handles the uncertainty as a probability distribution and guarantees the error bounds appropriately.

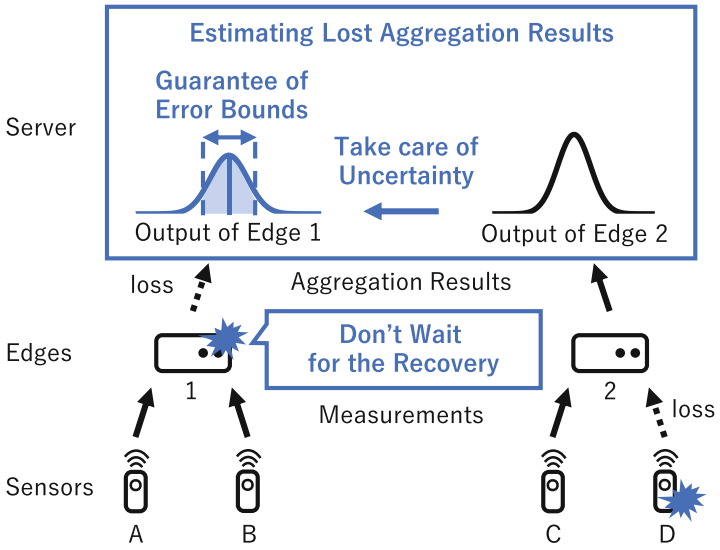


**Fig. 2.** Overview of the proposed approach

The rest of this paper is organized as follows. In Sect. 2, we discuss related work on fault tolerance in data stream processing systems. In Sect. 3, we explain the approximate aggregation at edges, which is the basis of our approach. In Sect. 4, we propose an approach to support approximate fault tolerance and provide a method to estimate the missing aggregation results taking care of the uncertainty. We evaluate the validity of our approach by experiments in Sect. 5, and we conclude the paper in Sect. 6.

## 2 Related Work

Many systems guarantee *at-most-once* semantics and *at-least-once* semantics [3, 13]. A system that guarantees at-most-once semantics only switches processing to the standby machines to recover from failures, but this approach can result in data loss. On the other hand, a system that guarantees at-least-once semantics reprocesses the lost data to recover from failures, but the same data can be processed multiple times. That is, these systems cannot guarantee the error bounds caused by failures.

Some stream processing systems, such as Flink [2] and Spark Streaming [6], guarantee *exactly-once* semantics. These systems recover from failure without error by restoring the last checkpoint of the internal states and reprocessing only lost data. This approach is suitable for cluster environments where the system can manage all resources centrally and assign the input data management task to the message queue.

However, for edge computing, there is a problem that the conventional approach needs to duplicate each edge. This duplication cost increases sharply with expansion in the system scale, unlike the cloud environment, which can manage all resources centrally. Besides, reprocessing lost data causes a delay because new inputs are generated one after another even during the reprocessing. We can suppress the delay by parallelizing failure recovery [16], but this approach also needs duplication.

AF-Stream supports approximate fault tolerance to address the trade-off between performance and accuracy [11]. This system mitigates the overhead by adaptively checkpointing while guarantees that the error upon failure is within the user-specified threshold. However, this system also requires duplication and is not suitable for edge computing environments.

## 3 Preliminaries

In this section, we explain the idea of approximate aggregation at edges [8], which is the basis of our approach.

In this paper, we assume that all $n$ sensors $\boldsymbol{X} = \{X_1, X_2, \ldots, X_n\}$ send measurements periodically and synchronously to the nearby edge. Let $\boldsymbol{x}^t = \langle x_1^t, x_2^t, \ldots, x_n^t \rangle$ be the true value of $\boldsymbol{X}$ at time $t \in \mathbb{N}^+$ and let $\boldsymbol{X}_i \subseteq \boldsymbol{X}$ be the sensor set assigned to edge $i$. If sensor failures or communication failures occur, edges cannot receive measurements from some sensors. In other word, at time $t$, edge $i$ only obtains the measurements $\boldsymbol{o}_i^t$ from sensors $\boldsymbol{O}_i^t \subseteq \boldsymbol{X}_i$. We assume that each sensor is assigned to only one edge, and the observed values are always equal to the true ones.

Edges divide the series of observations by time windows and aggregate them approximately for each time window. In this paper, we deal with two aggregation queries: *sum* and *average*. Let $t'$ be the start point of a time window and let $w$ be the window width. Edge $i$ calculates the aggregation result $\boldsymbol{Y}_X$ for each sensor $X \in \boldsymbol{X}_i$ from the series of observations $\boldsymbol{o}_i^{[t',t'+w)} = \langle \boldsymbol{o}_i^{t'}, \boldsymbol{o}_i^{t'+1}, \ldots, \boldsymbol{o}_i^{t'+w-1} \rangle$.

However what we really want is the aggregation result for $\boldsymbol{x}_i^{[t',t'+w)}$ not for $\boldsymbol{o}_i^{[t',t'+w)}$. It indicates that simply aggregating $\boldsymbol{o}_i^{[t',t'+w)}$ causes errors due to missing data and has no theoretical guarantee of the error bounds.

To solve this problem, we estimate missing data from observations. We utilize the correlation between sensors $\boldsymbol{X}$ modeled by a *multivariate Gaussian distribution* (hereafter, just Gaussian) $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$. Suppose that $\boldsymbol{\mu}$ and $\Sigma$ denote the mean vector and the covariance matrix of $\boldsymbol{X}$, respectively. For the measurements $\boldsymbol{o}_i^t$ of sensors $\boldsymbol{O}_i^t$, edge $i$ calculates the *posterior distribution* $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{X}_i^t|\boldsymbol{o}_i^t}, \Sigma_{\boldsymbol{X}_i^t|\boldsymbol{o}_i^t})$ for sensors $\boldsymbol{X}_i^t = \boldsymbol{X}_i \setminus \boldsymbol{O}_i^t$ [7]:

$$\boldsymbol{\mu}_{\boldsymbol{X}_i^t|\boldsymbol{o}_i^t} = \boldsymbol{\mu}_{\boldsymbol{X}_i^t} + \Sigma_{\boldsymbol{X}_i^t\boldsymbol{O}_i^t}\Sigma_{\boldsymbol{O}_i^t\boldsymbol{O}_i^t}^{-1}\left(\boldsymbol{o}_i^t - \boldsymbol{\mu}_{\boldsymbol{O}_i^t}\right) \text{ and} \tag{1}$$

$$\Sigma_{\boldsymbol{X}_i^t|\boldsymbol{o}_i^t} = \Sigma_{\boldsymbol{X}_i^t\boldsymbol{X}_i^t} - \Sigma_{\boldsymbol{X}_i^t\boldsymbol{O}_i^t}\Sigma_{\boldsymbol{O}_i^t\boldsymbol{O}_i^t}^{-1}\Sigma_{\boldsymbol{O}_i^t\boldsymbol{X}_i^t}. \tag{2}$$

The subscripts of each symbol indicate corresponding rows/columns in $\boldsymbol{\mu}$ and $\Sigma$. For instance, $\Sigma_{\boldsymbol{X}_i^t\boldsymbol{O}_i^t}$ is a sub matrix of $\Sigma$ formed by rows $\boldsymbol{X}_i^t$ and columns $\boldsymbol{O}_i^t$, which represents the covariance between $\boldsymbol{X}_i^t$ and $\boldsymbol{O}_i^t$.

Finally, we aggregate the estimated distributions in the window. In the following, we focus on the average query but we can handle the sum query in the same way. Since Gaussians are in the family of *stable distributions*, without temporal correlation, a Gaussian $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y}_{\boldsymbol{X}_i}}, \Sigma_{\boldsymbol{Y}_{\boldsymbol{X}_i}})$ for the average $\boldsymbol{Y}_{\boldsymbol{X}_i}$ of sensors $\boldsymbol{X}_i$ is represented as follows [8]:

$$\boldsymbol{\mu}_{\boldsymbol{Y}_{\boldsymbol{X}_i}} = \frac{1}{w}\left(\sum_{t\in[t',t'+w)} \boldsymbol{\mu}_{\boldsymbol{X}_i|\boldsymbol{o}_i^t}\right) \text{ and} \tag{3}$$

$$\Sigma_{\boldsymbol{Y}_{\boldsymbol{X}_i}} = \frac{1}{w^2}\left(\sum_{t\in[t',t'+w)} \Sigma_{\boldsymbol{X}_i|\boldsymbol{o}_i^t}\right). \tag{4}$$

Note that, for sensors $\boldsymbol{O}_i^t$, the mean vector is its measurements $\boldsymbol{o}_i^t$ and all the elements of the covariance matrix are 0.

This method theoretically guarantees the error bounds. Let $f(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma)$ be the probability density function for $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$. For an error bound $e$, we can calculate the probability that the average $Y_X$ of the sensor $X \in \boldsymbol{X}_i$ is within $e$ of the mean $\boldsymbol{\mu}_{Y_X}$:

$$P\left(Y_X \in [\boldsymbol{\mu}_{Y_X} - e, \boldsymbol{\mu}_{Y_X} + e]\right) = \int_{\boldsymbol{\mu}_{Y_X}-e}^{\boldsymbol{\mu}_{Y_X}+e} f\left(y \mid \boldsymbol{\mu}_{Y_X}, \Sigma_{Y_X}\right) dy. \tag{5}$$

That is, for a user-specified *confidence threshold* $\delta$, we can calculate the minimum error bound $e'$ that satisfies $\delta$ by solving:

$$P(Y_X \in [\mu_{Y_X} - e', \mu_{Y_X} + e']) = \delta. \tag{6}$$

This guarantees that the true value of $Y_X$ is within $\mu_{Y_X} \pm e'$ with the probability of $\delta$. Note that $\Sigma_{Y_X} = 0$ means that all data of $X$ are observed in the time window and the aggregation result has no error (i.e. $e' = 0$).

## 4 Proposed Approach

In this section, we propose our approach to support low-latency and high-availability fault tolerance without edge duplication. First, we explain how to guarantee approximate fault tolerance in our approach. Next, we provide a method to estimate the outputs of failed edges taking care of the uncertainty of the processing results.

### 4.1 Overview

We assume that we have $m$ edges that send the aggregation results to the server periodically and synchronously. Note that edge $i$ sends a Gaussian $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y}_{\boldsymbol{X}_i}}, \Sigma_{\boldsymbol{Y}_{\boldsymbol{X}_i}})$ as the aggregation result. For example, in Fig. 2, edge 2 calculates a bivariate Gaussian as the aggregation result for sensors C and D and sends it to the server. Then the server proceeds with the subsequent processing as soon as the processing results of all edges are available.

If a non-duplicated edge fails, the server cannot obtain the processing results during that time. In Fig. 2, due to the failure of edge 1, the server cannot obtain the aggregation result for sensors A and B and cannot proceed with the subsequent processing.

We solve this problem by estimating missing results using the correlation between sensors. Let $\boldsymbol{X}' \subseteq \boldsymbol{X}$ be the sensors assigned to failed edges (hereafter, referred to as *missing sensors*) and let $\boldsymbol{O} = \boldsymbol{X} \setminus \boldsymbol{X}'$ be the sensors assigned to remaining edges (hereafter, referred to as *observed sensors*). The server obtains a distribution $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y_O}}, \Sigma_{\boldsymbol{Y_O}})$ as the aggregation results $\boldsymbol{Y_O}$ for observed sensors and estimates a distribution $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y}_{\boldsymbol{X}'}}, \Sigma_{\boldsymbol{Y}_{\boldsymbol{X}'}})$ as the aggregation results $\boldsymbol{Y}_{\boldsymbol{X}'}$ for the missing sensors. Then the server continues the subsequent processing without waiting for the recovery of failed edges.

For example, in Fig. 2, the missing sensors are $\boldsymbol{X}' = \boldsymbol{X}_1 = \{X_A, X_B\}$ and the observed sensors are $\boldsymbol{O} = \boldsymbol{X}_2 = \{X_C, X_D\}$. The server can filter those whose aggregation result $Y_X$ for sensor $X \in \boldsymbol{X}'$ is less than a threshold $\theta$ in two steps: First, the server calculates the probability $P(Y_X < \theta)$ from the Gaussian $\mathcal{N}(\boldsymbol{\mu}_{Y_X}, \Sigma_{Y_X})$. Then the server checks whether this is greater than or equal to $\delta$.

In our approach, it is essential to take care of the uncertainty of processing results to estimate the outputs of failed edges accurately. For example, when estimating the output $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y}_{\boldsymbol{X}_1}}, \Sigma_{\boldsymbol{Y}_{\boldsymbol{X}_1}})$ of edge 1 from the output $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y}_{\boldsymbol{X}_2}}, \Sigma_{\boldsymbol{Y}_{\boldsymbol{X}_2}})$ of edge 2 in Fig. 2, we can consider a method that uses the posterior probability distribution given the mean vector $\boldsymbol{\mu}_{\boldsymbol{Y}_{\boldsymbol{X}_2}}$ as described in Sect. 3. However, this method does not use the covariance matrix $\Sigma_{\boldsymbol{Y}_{\boldsymbol{X}_2}}$, which represents the uncertainty of the output of edge 2, and results in an underestimation of the error bounds.

### 4.2 Aggregation Result Estimation

If we can model the correlation between sensors $\boldsymbol{X}$ by a Gaussian $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$, we can also model the correlation between the aggregation results $\boldsymbol{Y_X}$ by a Gaussian.

Consider an average query for a time window with the window width $w$. From the reproductive property of Gaussian, we can model the correlation between $\boldsymbol{Y_X}$ by a Gaussian $\mathcal{N}(\boldsymbol{\mu}', \Sigma')$, whose mean vector and covariance matrix are:

$$\boldsymbol{\mu}' = \frac{1}{w}\left(\sum_{t\in[t',t'+w)} \boldsymbol{\mu}^t\right) = \boldsymbol{\mu} \text{ and} \tag{7}$$

$$\Sigma' = \frac{1}{w^2}\left(\sum_{t\in[t',t'+w)} \Sigma^t\right) = \frac{1}{w}\Sigma. \tag{8}$$

If the server knows that the aggregation results $\boldsymbol{Y_O}$ are equal to $\boldsymbol{y}$, the server can calculate the posterior probability distribution $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y_{X'}}|\boldsymbol{y}}, \Sigma_{\boldsymbol{Y_{X'}}|\boldsymbol{y}})$ as the estimated aggregation results $\boldsymbol{Y_{X'}}$:

$$\boldsymbol{\mu}_{\boldsymbol{Y_{X'}}|\boldsymbol{y}} = \boldsymbol{\mu}'_{\boldsymbol{X'}} + \Sigma'_{\boldsymbol{X'O}}\left(\Sigma'_{\boldsymbol{OO}}\right)^{-1}\left(\boldsymbol{y} - \boldsymbol{\mu}'_{\boldsymbol{O}}\right) \text{ and} \tag{9}$$

$$\Sigma_{\boldsymbol{Y_{X'}}|\boldsymbol{y}} = \Sigma'_{\boldsymbol{X'X'}} - \Sigma'_{\boldsymbol{X'O}}\left(\Sigma'_{\boldsymbol{OO}}\right)^{-1}\Sigma'_{\boldsymbol{OX'}}. \tag{10}$$

However, the server cannot know the exact value $\boldsymbol{y}$ because edges aggregate sensor data approximately.

In our method, the server estimates the aggregation results $\boldsymbol{Y_{X'}}$ using the Gaussian $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y_O}}, \Sigma_{\boldsymbol{Y_O}})$ that represents the uncertainty of $\boldsymbol{Y_O}$. That is, the server calculates the probability that $\boldsymbol{Y_O}$ is equal to $\boldsymbol{y}$ from $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y_O}}, \Sigma_{\boldsymbol{Y_O}})$ and marginalizes it:

$$P\left(\boldsymbol{Y_{X'}}\right) = \int P\left(\boldsymbol{Y_O} = \boldsymbol{y}\right) P\left(\boldsymbol{Y_{X'}} \mid \boldsymbol{Y_O} = \boldsymbol{y}\right) d\boldsymbol{y} \tag{11}$$

$$= \int f\left(\boldsymbol{y} \mid \boldsymbol{\mu}_{\boldsymbol{Y_O}}, \Sigma_{\boldsymbol{Y_O}}\right) f\left(\boldsymbol{y}' \mid \boldsymbol{\mu}_{\boldsymbol{Y_{X'}}|\boldsymbol{y}}, \Sigma_{\boldsymbol{Y_{X'}}|\boldsymbol{y}}\right) d\boldsymbol{y}. \tag{12}$$

Note that, some edges aggregate measurements of a part of sensors $\boldsymbol{O}$ separately, so that the distribution $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y_O}}, \Sigma_{\boldsymbol{Y_O}})$ of $\boldsymbol{Y_O}$ is a combination of their outputs. In this paper, since edge $i$ cannot get the observations of sensors $\boldsymbol{X}_j$ assigned to another edge $j$, we consider all of the covariances between $\boldsymbol{Y_{X_i}}$ and $\boldsymbol{Y_{X_j}}$ in $\Sigma_{\boldsymbol{Y_O}}$ as 0.

Calculating Eq. (12), we get a Gaussian $\mathcal{N}(\boldsymbol{\mu}'_{\boldsymbol{Y_{X'}}}, \Sigma'_{\boldsymbol{Y_{X'}}})$, whose mean vector and covariance matrix are:

$$\boldsymbol{\mu}'_{\boldsymbol{Y_{X'}}} = \boldsymbol{\mu}'_{\boldsymbol{X'}} + \Sigma'_{\boldsymbol{X'O}}\left(\Sigma'_{\boldsymbol{OO}}\right)^{-1}\left(\boldsymbol{\mu}_{\boldsymbol{Y_O}} - \boldsymbol{\mu}'_{\boldsymbol{O}}\right) \text{ and} \tag{13}$$

$$\Sigma'_{\boldsymbol{Y_{X'}}} = \Sigma'_{\boldsymbol{X'X'}} - \Sigma'_{\boldsymbol{X'O}}\left(\Sigma'_{\boldsymbol{OO}}\right)^{-1}\Sigma'_{\boldsymbol{OX'}} + \Sigma'_{\boldsymbol{X'O}}\left(\Sigma'_{\boldsymbol{OO}}\right)^{-1}\Sigma_{\boldsymbol{Y_O}}\left(\Sigma'_{\boldsymbol{OO}}\right)^{-1}\Sigma'_{\boldsymbol{OX'}}. \tag{14}$$

Note that these are similar to the posterior probability distribution given the mean vector $\boldsymbol{\mu}_{\boldsymbol{Y_O}}$ as the aggregation results: the mean vector is equal to Eq. (13) and the covariance matrix is equal to the first and second terms of Eq. (14).

The difference is the third term of Eq. (14) which is the correction term of the covariance matrix based on the uncertainty of processing results. It indicates that, unlike simply estimating the result of $\boldsymbol{Y_{X'}}$ from the mean vector $\boldsymbol{\mu_{Y_O}}$, our method corrects the covariance matrix $\Sigma'_{\boldsymbol{Y_{X'}}}$ taking care of the uncertainty to predict the error bounds accurately.

## 5 Experimental Evaluation

In this section, we demonstrate the validity of the proposed method by experiments. We explain the experimental settings and then describe the experimental results in detail. Note that we omit the results of the sum query because it has the same tendency as the average query.

### 5.1 Experimental Settings

We use the following two datasets to evaluate the validity of our method.

**Real.** We measure the temperature in our laboratory with 24 environmental sensors 2JCIE-BL [1]. This dataset has tuples for 24 days at 1-minute intervals. We use the data for the first 16 days for model training and the rest for the evaluation. Note that pre-test using `mvnormtest` [4] revealed that this data does not follow the multivariate Gaussian.

**Synthetic.** We prepare synthetic data to verify the validity of the theoretical error guarantee in our method. The dataset has 11520 tuples correspond to 8 days, which follow the Gaussian obtained from the training data. We use `rmvnorm` function [5] that generates an $n$-dimensional random number vector that follows the $n$-dimensional Gaussian given as input. Note that each tuple has no temporal correlation because we generate tuples independently.

In this experiment, we assume a simple failure scenario and prepare test data in three steps: First, we randomly delete part of the measurements at the drop rate $r$ as data lost due to temporary communication failures between sensors and edges. Next, we split this partially deleted data by the window width $w$ and calculate the multivariate Gaussian for the aggregation results as described in Sect. 3. Finally, we randomly select $m$ failed edges and delete these aggregation results as data lost due to edge failures.

### 5.2 Methodology

We evaluate the validity of our approach from the perspective of theoretical error guarantee. In our approach, the server estimates the missing aggregation result $Y_X$ for sensor $X$ as a Gaussian $\mathcal{N}(\mu_{Y_X}, \Sigma_{Y_X})$ and guarantees its error bounds $e'$ for a user-specified confidence threshold $\delta$. It indicates that the true value of $Y_X$ is theoretically within $\mu_{Y_X} \pm e'$ with the probability of $\delta$. Therefore, we consider the aggregation result for the lossless dataset as the true value and define the

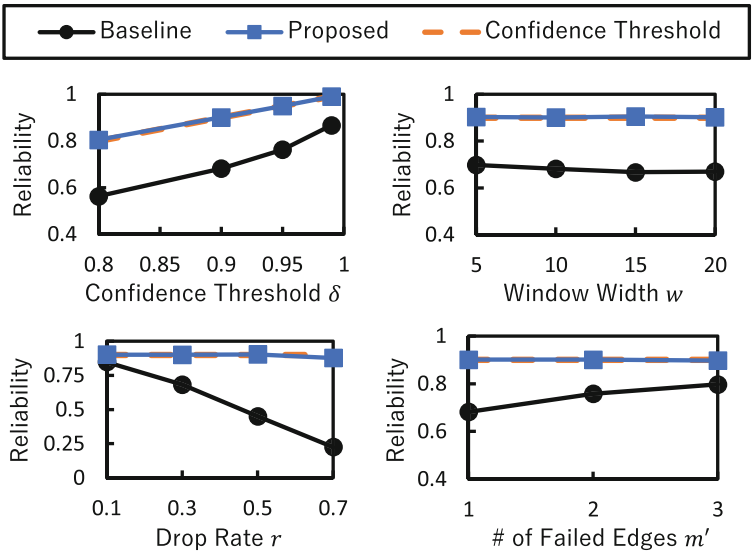


**Fig. 3.** Experimental results for synthetic data

*reliability* as the probability that the true value exists within $\mu_{Y_X} \pm e'$. When the reliability satisfies $\delta$, we can consider the error guarantee of our approach to be valid.

We evaluate the reliability by varying each parameter: confidence threshold $\delta$, window width $w$, drop rate $r$, and the number of failed edges $m'$. Note that these default values are $\delta = 0.9$, $w = 10$, $r = 0.3$, and $m' = 1$, respectively. We compare our method with a method that estimates missing results not taking care of the uncertainty (i.e. calculating Eq. (9) and Eq. (10) with $\boldsymbol{y} = \boldsymbol{\mu}_{\boldsymbol{Y_O}}$). We refer to this method as *baseline*.

### 5.3 Results

We show the experimental results for the synthetic data in Fig. 3. The dotted lines in the figures show the confidence threshold $\delta$. The reliability of the baseline is always below $\delta$, whereas our method satisfies $\delta$ for all parameter settings. These graphs show that the uncertainty of aggregation results affects the estimation of error bounds and our method can recover missing aggregation results reliably from partial observations.

Consider the tendency of reliability change for each parameter. We see that the difference in reliability between the baseline and our method increases as the error rate $r$ increases. It is considered that an increase in missing tuples causes an increase in the uncertainty of processing results of each edge, which results in a lack of reliability of the baseline. The difference also increases as the number of failed edges $m'$ decreases. A decrease in failed edges means an increase in the observed processing results with uncertainty. We find that our method

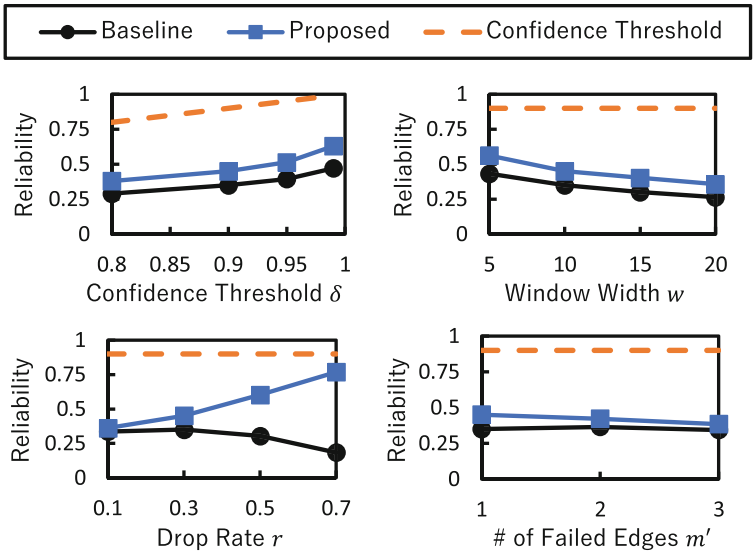


**Fig. 4.** Experimental results for real data

is particularly effective in these scenarios where the impact of the uncertainty increases.

On the other hand, as shown in Fig. 4, our method does not satisfy the confidence threshold for real data. We found that this is due to temperature changes over time. To solve this problem, we have to extend our method to handle temporal correlations. We are currently considering an approach to update the model by applying the Kalman filter [9]. Also, we should pay attention to the difference in daily trends due to human activity. For example, the temperature is kept by the air conditioner when someone is in the room, whereas the temperature decreases gradually when the room is vacant. We can address this problem by simply learning a different model for each situation.

## 6 Conclusions

We proposed an approach to guarantee approximate fault tolerance for edge computing environments. In our approach, the server estimates missing aggregation results using the correlation between sensors to continue the processing without waiting for the recovery of failed edges. We also provided a method to estimate missing aggregation results taking care of the uncertainty of observed processing results. The experimental results demonstrated that the importance of handling uncertainty and the validity of our method. However, our method could not estimate error bounds accurately for real data. To solve this problem, we are currently considering extending our method to handle temporal correlations.

## References


1. 2JCIE-BL Environment Sensor | OMRON - Americas. https://components.omron.com/product-detail?partId=73064. Accessed 20 Apr 2021
2. Apache Flink: Stateful Computations over Data Streams. https://flink.apache.org/. Accessed 20 Apr 2021
3. Apache Storm. https://storm.apache.org/. Accessed 20 Apr 2021
4. CRAN - package mvnormtest. https://CRAN.R-project.org/package=mvnormtest. Accessed 20 Apr 2021
5. Mvnorm function | R Documentation. https://www.rdocumentation.org/packages/mvtnorm/versions/1.0-11/topics/Mvnorm. Accessed 20 Apr 2021
6. Spark Streaming | Apache Spark. https://spark.apache.org/streaming/. Accessed 20 Apr 2021
7. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006). https://doi.org/10.1007/978-0-387-45528-0
8. Daiki, T., Kento, S., Yoshiharu, I.: Approximate streaming aggregation with low-latency and high-reliability for edge computing. IEICE Trans. Inf. Syst. **J104-D**(5), 463–475 (2021). (in Japanese)
9. Deshpande, A., Guestrin, C., Madden, S.R., Hellerstein, J.M., Hong, W.: Model-based approximate querying in sensor networks. VLDB J. **14**(4), 417–443 (2005)
10. Enders, C.K.: Applied Missing Data Analysis. Guilford Press, New York (2010)
11. Huang, Q., Lee, P.P.C.: Toward high-performance distributed stream processing via approximate fault tolerance. Proc. VLDB **10**(3), 73–84 (2016)
12. Hwang, J.H., Balazinska, M., Rasin, A., Çetintemel, U., Stonebraker, M., Zdonik, S.: High-availability algorithms for distributed stream processing. In: Proceedings of ICDE, pp. 779–790, April 2005
13. Neumeyer, L., Robbins, B., Nair, A., Kesari, A.: S4: Distributed stream computing platform. In: 2010 IEEE International Conference on Data Mining Workshops, pp. 170–177, January 2010
14. Rasmussen, E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press, Cambridge (2006)
15. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
16. Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., Stoica, I.: Discretized streams: a fault-tolerant model for scalable stream processing. Technical report, California University of Berkeley, Department of Electrical Engineering and Computer Science, (2012)