Research Article

Design of Management Platform Architecture and Key Algorithm for Massive Monitoring Big Data

Wen Liu ^{1,2,3}, Shanshan Li,³ Jiangru Pan,¹ Lijun Xu,^{1,2} Zheng Gu,¹ and Ling Hai¹

¹School of Control Engineering, Xinjiang Institute of Engineering, Ürümqi 830000, China

²Xinjiang Sunshine Diantong Technology Co., Ltd, Ürümqi 830000, China

³College of Computer Science and Technology, Xinjiang Normal University, Ürümqi 830000, China

Correspondence should be addressed to Wen Liu; 627952@qq.com

Received 11 August 2021; Revised 8 September 2021; Accepted 17 September 2021; Published 28 September 2021

Academic Editor: Yong Deng

Copyright © 2021 Wen Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the construction and development of industrial informatization, industrial big data has become a trend within the smart industry. To obtain valuable information on massive data, achieving the acquisition, storage, analysis, and mining is becoming an important area of research. Focusing on the application requirements for industrial fields, we propose a data acquisition and analysis system based on the NB-IoT for industrial applications. The system is an integrated system that includes sensor data acquisition, data transmission, data storage, and analysis mining. In this study, we mainly focused on the use of the NB-IoT network to collect and transmit real-time data for sensors. First, for the long time series (e.g., if we collect the data streams for one year for the sensor with a frequency of 1 Hz, the length of the series will reach 10^7). Then, we propose DSCS-LTS, a distributed storage and calculation model, and CCCA-LTS, an algorithm for the correlation coefficient of long time series in a distributed environment. Third, we propose a granularity selection algorithm and query process logic for visualization. We tested the platform in our laboratory and an automated production line for one year, and the experimental results using real data sets show that our approach is effective and scalable, can achieve efficient data management, and provide the basis for intelligent enterprise decision-making.

1. Introduction

With the rapid development of “interconnected” and “intelligent” industries, industrial big data has become the focus of current research. For all types of real-time monitoring of data in the industrial field, the technology of collection, storage, and analysis, to enable deep mining, is facing great challenges. The construction and development of the “interconnected” industry and “intelligent” industry require a large amount of basic data. These data are not only to stay in the collection stage but also to carry out deep storage, analysis, and mining, for example, real-time analysis of industrial field data, alarm of abnormal data, supervision of every process in the production process, and intelligent decision-making. This series of problems is becoming an important topic in industrial informatization research.

With the development of the Internet of Things technology, the traditional big data acquisition platforms have been

unable to adapt to the growing mass of data. However, the scheme of “centralized collection and centralized management” provides an effective solution for massive data management in the industrial field. In this paper, this specific project is an example: a large tomato sauce factory production line, collecting all kinds of monitoring data (such as sterilization temperature, tank pressure, pipeline flow rate, and motor power), real-time data through the network transmission to the data management platform, for massive data design distributed storage and analysis methods, realize data depth mining and visualization, and provide the basis for decision-making.

Based on the current situation of large tomato sauce production lines, this project designs an integrated platform for data collection, analysis, mining, and visualization. We propose a distributed storage, calculation model DSCS-LTS, correlation coefficient estimation method CCCA-LTS for massive, long time series data, and propose a data

visualization method that addresses the key problems related to the management of massive monitoring data.

2. Related Research

2.1. Related Research Work

2.1.1. Research on Time Series Data Management Platform.

In recent years, as the time series data management system (TSDB) has been developing rapidly, many time series management systems have emerged. Popular open-source time series data management systems include InfluxDB [1] and OpentsDB [2]. InfluxDB defines the field types and statistical queries for time series. However, it does not support complex queries, such as a similarity search. OpentsDB is based on the HBase-distributed time series management system, which supports simple statistical queries on time series. Recently, Tsinghua University in China developed IoTDB [3], which has become the Apache incubation project. The time series storage structure is similar to that of Parquet [4] and supports distributed computing frameworks such as MapReduce and Spark. There are also many commercial time series databases, such as the domestic T-engine. However, these systems only support basic aggregation queries but do not support association analysis, similarity queries, and other functions.

2.1.2. Similarity Query Technology for Time Series. Time series similarity query technology has been studied for approximately 20 years. In 1993, Lomet first proposed the research problem of similarity queries for time series databases [5] and discrete Fourier transforms to reduce the dimension of time series. Second, the R-tree was used to build the index processing method. Third, the main research uses other dimension reduction technologies, which follow the routine of dimensionality reduction before index building, including discrete wavelet transform [6], APCA [7], and SAX [8]. In 2013, with respect to the complete sequence similarity query, VLDB [9] proposed a mechanism combining dimensionality reduction and index construction. In 2014, Zoumpatianos et al. proposed refining the index mechanism in the query stage, which shortened the waiting time for index construction [10]. In 2012, Rakthanmanon et al. proposed a UCR suite algorithm for subsequence similarity query [11]. Their approach supports standardized subsequence similarity queries. However, this approach cannot build an index and scan the entire sequence. In 2019, the VLDB paper proposed a variable-length standardized subsequence similarity query algorithm [12]. In 2016, the Harbin Institute Technology team proposed a set-based approximate query algorithm for time series at the SIGMOD conference [13]. The team from the University of Chinese Academy of Sciences and the State Grid Electric Power Research Institute proposed a multidimensional query system DGIndex for smart grid data [14], and the Beihang team proposed an approximate representation and query algorithm for trajectory time series [15]. Their approach supports one-dimensional, few-dimensional time series or specific aggregate functions and cannot achieve aggregate query processing on a large-scale time series. In summary,

current scientific computing, the Internet of Things, and intelligent manufacturing have become research hotspots globally. The MengXiaofeng team of Renmin University of China proposed a scientific big data management framework, suitable for the entire life cycle of scientific data management and analyzed the key technologies in the scientific big data management system [16].

2.1.3. Correlation Coefficient Calculation of Time Series. In the past 20 years, there have been many mining algorithms and query algorithms for time series data. Time series mining algorithms include classification, clustering, outlier detection, and motif mining [17]. Time series query algorithms include approximate queries [18], aggregate queries [19], and range queries [20, 21]. The above algorithm can only be used in a single machine environment and is not suitable for processing massive, long time series data (for example, a sensor with a frequency of 1 Hz can continuously generate 10^7 data for one year).

Computing correlation coefficients for long time series in a distributed environment has the following problems: (1) distributed calculations cannot be performed. Although the Euclidean distance can be computed in a distributed fashion, it requires the mean and standard deviation of the entire sequence to compute the correlation coefficient, so it cannot be computed in a distributed fashion; (2) when a query sequence is long, it requires extensive I/O and network costs, thus causing delays, and cannot be used in interactive query applications. To solve these problems, we propose a method to estimate the correlation coefficients of two sequences on HBase and design a fast estimation method, CCCA-LTS, for the upper and lower bounds of the correlation coefficients. The HBase algorithm iteratively estimates the correlation coefficients.

2.2. Research Contents. This study conducts research, based on a mass monitoring data management platform, to provide an intelligent decision-making and control basis for enterprise production. The specific work is as follows: first, we designed a collection of various monitoring data collection terminals and used the NB-IoT network to transmit the data to the management platform. Second, according to the characteristics of the data, we designed the distributed storage and calculation model DSCS-LTS, which realizes the efficient storage of long-term sequences. Third, in order to calculate the correlation between the series, we designed the correlation coefficient estimation method CCCA-LTS for long-term series data; and fourth, we designed the granular selection algorithm and query process, which logically realizes the visualization of the data. The overall system architecture and data acquisition terminal design are shown in Figures 1 and 2, respectively.

The overall framework of the system is mainly composed of a collection equipment layer, a communication channel layer, and a master station layer. The acquisition device layer realizes the acquisition, processing, and real-time monitoring of monitoring data; the communication channel layer transmits the massive real-time data stream to the master station layer; the master station layer completes the data

FIGURE 1: System framework diagram.

FIGURE 2: Structure diagram of acquisition terminal.

stream processing, storage, data analysis, mining, and visualization.

According to the data characteristics of sensors and instruments in the industrial field, we designed the data acquisition and data transmission terminal (including the latest NB-IoT module), as shown in Figure 2. The terminal uses an STM32 single-chip microcomputer as a processor, connects field sensors, meters, etc., through the acquisition module, and then transmits the collected data to the data management platform through the NB-IoT network.

The master station layer regards the construction of the sensor data management system as the overall goal and conducts research on actual business needs and data characteristics, as well as from platform construction, data collection, data analysis, data mining, and other levels.

2.3. Key Technology of the System

2.3.1. Prerequisite Knowledge

Definition 1 (time series). The time series can be expressed as $S = ((t_1, s_1), (t_2, s_2), \dots, (t_n, s_n))$. n is the total length of S , t_i is the timestamp, and s_i is the value of t_i and $(1 \leq i \leq n)$.

Definition 2 (equally interval time series). The equal interval time series refers to the set S that arranges the indicators of a certain phenomenon in time sequences and equal time intervals, denoted as $S = (s_1, s_2, \dots, s_n)$.

For ease of description, the time series presented in this article are all equally spaced time series, and this algorithm is also suitable for nonequally spaced time series.

Definition 3 (subsequence). $S_{i:l} = (s_i, s_{i+1}, \dots, s_{i+l-1})$ is a subsequence of S . The relationship between i and l is $1 \leq i \leq n-l$.

Definition 4 (Pearson correlation coefficient). The lengths of the time series $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$ are both n . The Pearson correlation coefficient was calculated as follows:

$$\rho(X, Y) = \frac{1}{n} \sum_{i=1}^n \left(\frac{x_i - \mu_x}{\sigma_x} \right) \left(\frac{y_i - \mu_y}{\sigma_y} \right). \quad (1)$$

μ_x and μ_y are the mean values of X and Y , respectively, $\mu_x = (1/n) \sum_{i=1}^n x_i$. σ_x and σ_y are the standard deviations of X and Y , respectively, $\sigma_x = \sqrt{\sum_{i=1}^n (x_i)^2 / n - (\mu_x)^2}$.

Problem definition: the time series database SS is stored in a distributed architecture HDFS or HBase. Two subsequences X and Y in database SS calculate whether the Pearson coefficient $\rho(X_{i:l}, Y_{i:l}) \geq \varepsilon$ is valid, where i is any integer, l is the subsequence length, and ε is the correlation coefficient threshold. In this study, the query window is set to $(i, i+1, i+2, \dots, i+l-1)$.

2.4. Distributed Storage Scheme and Computing Model

2.4.1. Distributed Storage Solution Based on HBase. At present, many applications use the distributed storage architecture HDFS and HBase to store massive amounts of time series data. Representative platforms include OpenTSDB [1] and TempoIQ [17]. There are two schemes on the HBase time series storage method.

Scheme 1: Figure 3 shows the first storage solution of HBase. Based on storage method 1, we can access the value of any sequence at any timestamp.

Scheme 2: Figure 4 shows the second storage solution of HBase. The scheme is based on Scheme 1, storing a subsequence consisting of a continuous period value.

This paper proposed an algorithm that satisfies the above two schemes.

2.4.2. Distributed Storage and Computing Model DSCS-LTS. To improve the generalization, we use a distributed storage model and computing model DSCS-LTS (distributed storage and calculation scheme for long time series) for the above

FIGURE 3: Storage method 1.

FIGURE 4: Storage method 2.

storage scheme, which is a long time series for massive monitoring distributed storage and calculation model of sequence data.

Distributed storage: L storage nodes $\{N_1, N_2, \dots, N_L\}$ in a distributed environment. Time series data in the database SS are divided into several disjoint subsequences and store them into L sequence nodes. The subsequence is $S_i = (s_{(i-1) \cdot w + 1}, s_{(i-1) \cdot w + 2}, \dots, s_{i \cdot w})$, $1 \leq i \leq \lceil n/w \rceil$ where w represents the time dimension.

The subsequence database stored by node N_j is represented as SS_j , and the sequence S_i subsequence S is represented as $S_i \in SS_j$ if it is stored within a node N_j . When the length of a subsequence equals 1 ($w = 1$), only one value of a time series exists in each row, so Scheme 1 is meaningless.

Distributed computing: as shown in Figure 5, it is a distributed computing process. There are L several computing nodes $\{N_1, N_2, \dots, N_L\}$. All L nodes have storage and computing capabilities. N_0 is the query driving node, which comprehensively handles all o_i results.

2.5. CCCA-LTS Calculation Method of the Long Time Series Correlation Coefficient in a Distributed Environment

2.5.1. CCCA-LTS Algorithm. CCCA-LTS (correlation coefficient calculation algorithm for long time series) is a Pearson correlation coefficient estimation method in a distributed environment. In the correlation coefficient estimation algorithm, we assume that $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$ are complete subsequences, and the query window is $(1, 2, \dots, n)$. The CCCA-LTS algorithm can be directly extended to any query window.

As shown in Figure 6 (step 1 and step 2), the sequences X and Y are divided into six subsequences (step 1). Then, six subsequences are distributed to the data nodes (step 2). A simple way is to transmit all subsequences to N_0 for calculation, which will increase network transmission. To effectively reduce the cost of network transmission, this study proposes the CCCA-LTS algorithm.

FIGURE 5: Distributed computing framework.

FIGURE 6: CCCA-LTS algorithm.

The core of the CCCA-LTS algorithm is illustrated in Figure 6 (step 3). All L nodes need to calculate the mean and standard deviation of the subsequence stored in the node (step 3). For example, in Figure 6, the subsequence X_1 is stored at node N_1 , and node N_1 calculates μ_1^X and σ_1^X , $\mu_1^X = (1/w) \sum_{i=1}^w x_i$, and $\sigma_1^X = \sqrt{(\sum_{i=1}^w (x_i)^2 / w) - (\mu_1^X)^2}$. Then, the values of the calculated results are transmitted to the node N_0 . N_0 estimated the correlation coefficient according to these values. Next, the CCCA-LTS algorithm was introduced in detail.

2.5.2. Relationship between Correlation Coefficient and Euclidean Distance. We first provide the estimation formulas, based on the upper and lower bounds in Figure 6 (step 3). The normalized sequences of X and Y are $\hat{X}_i = (x_i - \mu_X) / \sigma_X$ and $\hat{Y} = \{\hat{y}_i\}_{1 \leq i \leq n}$. The relationship between the correlation coefficient of X and Y and the Euclidean distance:

$$\rho(X, Y) = 1 - \frac{1}{2n} d^2(\hat{X}, \hat{Y}). \quad (2)$$

According to Equation (2), the formula $d^2(\hat{X}, \hat{Y}) \leq 2n(1 - \varepsilon)$ can be used to estimate the upper and lower bounds.

2.5.3. Correlation Coefficient Estimation Method. $\hat{X}$ and $\hat{Y}$ are standardized time series. We first introduce the

approximate representation method EAPCA of the time series, proposed in Reference [9]. Then, we represent $R(\hat{X})$ and $R(\hat{Y})$ according to the EAPCA of $\hat{X}$ and $\hat{Y}$. Finally, we give estimates of $d(\hat{X}, \hat{Y})$.

EAPCA first divides the series S into $S = (S_1, S_2, \dots, S_m)$; the arbitrary segment is $S_j = (s_{r_{j-1}+1}, s_{r_{j-1}+2}, \dots, s_{r_j})$, ($1 \leq j \leq m$, $1 \leq r_1 < r_2 < \dots < r_m \leq n$). The EAPCA of S is denoted as $R(S) = ((\mu_1^S, \sigma_1^S, r_1), (\mu_2^S, \sigma_2^S, r_2), \dots, (\mu_m^S, \sigma_m^S, r_m))$, where X and Y are the mean and standard deviation of S_j , respectively. We denote $\hat{X}$ and $\hat{Y}$ as $R(\hat{X})$ and $R(\hat{Y})$, as follows:

$$\begin{aligned} R(\hat{X}) &= ((\mu_1^{\hat{X}}, \sigma_1^{\hat{X}}, r_1), (\mu_2^{\hat{X}}, \sigma_2^{\hat{X}}, r_2), \dots, (\mu_m^{\hat{X}}, \sigma_m^{\hat{X}}, r_m)), \\ R(\hat{Y}) &= ((\mu_1^{\hat{Y}}, \sigma_1^{\hat{Y}}, r_1), (\mu_2^{\hat{Y}}, \sigma_2^{\hat{Y}}, r_2), \dots, (\mu_m^{\hat{Y}}, \sigma_m^{\hat{Y}}, r_m)). \end{aligned} \quad (3)$$

According to [9], we obtain the boundary of the Euclidean distance of $\hat{X}$ and $\hat{Y}$ as follows:

$$d(\hat{x}, \hat{y}) \geq \sqrt{\sum_{i=1}^m (r_i - r_{i-1}) \left[(\mu_i^{X\wedge} - \mu_i^{Y\wedge})^2 + (\sigma_i^{X\wedge} - \sigma_i^{Y\wedge})^2 \right]}, \quad (4)$$

$$d(\hat{x}, \hat{y}) \leq \sqrt{\sum_{i=1}^m (r_i - r_{i-1}) \left[(\mu_i^{X\wedge} - \mu_i^{Y\wedge})^2 + (\sigma_i^{X\wedge} + \sigma_i^{Y\wedge})^2 \right]}. \quad (5)$$

Through the above analysis, we summarize as follows:

- (i) If $\sqrt{2n(1-\varepsilon)}$ is more than $d(\hat{X}, \hat{Y})$ (upper bond), then there must be $\rho(X, Y) > \varepsilon$
- (ii) If $\sqrt{2n(1-\varepsilon)}$ is less than $d(\hat{X}, \hat{Y})$ (lower bound), then $\rho(X, Y) < \varepsilon$ must exist
- (iii) If neither of the above two situations is true, it is impossible to judge whether $\rho(X, Y) > \varepsilon$ is true

2.5.4. Distributed Estimation Methods. In a distributed environment, as shown in Figure 6 (step 2), we cannot calculate the mean and standard deviation of the standardized subsequence $\hat{X}$. The calculation was the same for the standardized subsequence $\hat{Y}$.

This paper proposes a new method for estimating the standardized mean and standard deviation in a distributed environment. As shown in Figure 6 (step 3), each node first calculates the $\mu_i^X, \delta_i^X, \mu_i^Y$, and δ_i^Y and then enters the result into N_0 . These mean and standard deviation values were used to estimate the upper and lower bounds, respectively. We provide estimation methods for $\mu_i^{\hat{X}}$ and $\sigma_i^{\hat{X}}$, $1 \leq i \leq m$. We can estimate the complete series by Equations (5)–(8), where μ_i^X and σ_i^X are the mean and standard deviation of the known subsequence; μ^X and δ^X are the overall mean

and standard deviation, respectively. The proofs of Equations (6)–(9) are given in Reference [2] and will not be repeated here.

$$\mu^X = \frac{\sum_i (r_i - r_{i-1}) \mu_i^X}{n}, \quad (6)$$

$$\sigma^X = \sqrt{\frac{\sum_{i=1}^m (r_i - r_{i-1}) \left((\mu_i^X)^2 + (\sigma_i^X)^2 \right)}{n} - (\mu^X)^2}, \quad (7)$$

$$\mu_i^{\hat{X}} = \frac{\mu_i^X}{\sigma^X} - \frac{\mu^X}{\sigma^X}, \quad (8)$$

$$\sigma_i^{\hat{X}} = \frac{\sigma_i^X}{\sigma^X}. \quad (9)$$

Using Equations (2), (4), (5), (6), (7), (8), and (9), we can calculate the correlation coefficient.

2.5.5. CCCA-LTS Algorithm. This section discusses two problems with the CCCA-LTS algorithm. The first problem is that in the previous description, we assume that the query window is the entire window, expressed as $(1, 2, \dots, n)$. The actual query window boundary does not necessarily fall on the boundary of the subsequence and may fall within the subsequence. When the query window is within the internal sequence, the query sequence is necessary to read out the window border, and there is a need to calculate the mean and standard query portion difference within the window.

The second question is whether there are three possible outcomes. When the third result appears, that is, when it is impossible to judge, we need to estimate based on the more fine-grained mean and standard deviation. At this point, we need a second or even third round of calculation. In the first round, the subsequence length is w , the mean and standard deviation are calculated, and N_0 is returned for comprehensive calculation and judgment. When the judgment result is the third type, we need to the second round of calculation; that is, we calculate the subsequence with length $w/2$ and then return N_0 for comprehensive calculation and judgment. If the judgment result is of the first or second type, it stops. Otherwise, the third round is performed, that is, e the mean and standard deviation of the subsequence with length $w/4$ are calculated. Thus, the judgment accuracy meets the needs.

CCCA-LTS algorithm analysis: the CCCA-LTS algorithm is a multiround algorithm. If only one round of calculation is required, the CCCA-LTS algorithm has the same I/O overhead as the direct calculation method. As only the mean value and standard deviation of the subsequence are transmitted, the network has a significantly reduced transmission cost. If multiple rounds of calculation are required, the query window sequence requires multiple reads, thus causing more I/O overhead.

2.6. Visualization Technology. This section focuses on two issues: granularity data visualization algorithms and query process technology. The selected size has a significant influence on the response time of the query. When the particle size is too small, a larger network transmission is required,

Algorithm 1: Select granularity (X, start time, end time, M)

Input: X time series ID, start and end time, client can display the maximum amount of data

Output: < size of granularity > Calculate the size of different granularity data of X Time Series in the historical statistics table from start time to end time;

Sort the amount of data in step 1 in descending order;

Select the first granularity that is smaller than the maximum amount of data that can be displayed on the client.

ALGORITHM 1: Granularity selection algorithm.

and the resulting query response time is too high; therefore, a large amount of data is not stored in the client memory. When the granularity is large, the amount of data transmitted to the user is small, which cannot accurately represent the trend of the original time series.

In the query process, the user sets a good time interval and a data channel. In general, a user should choose an appropriate statistic. If the user is not explicitly given, the default is the appropriate statistical median. The statistics currently supported include mean and median. When a user requests a query data for a certain period of time, he/she can select a period of interest in the front end of the data to view more detailed data trends and other information during this period.

2.6.1. Granularity Algorithm. Select the required particle size to be determined, according to the size of the data. If the granularity is too large, the user query response time will be smaller, but the amount of data returned to the client will be smaller, thereby increasing the error. If the granularity is too small, the amount of data returned to the client will be too large. Although the error is small, the user's query response time is long.

To improve the response time of the query, we design a historical data statistics table. The historical data statistics table contains statistical data of different granularities, such as the maximum and minimum values of a certain time series within one hour. The existing granularities are day, hour, minute, and second. It is a challenge not only to find a good representation of the original time series trends in large amounts of historical data in tables but also to avoid the transmission of data to the client that is too large. The idea to solve this problem is to calculate the amount of data of different granularities according to the frequency of the time series and then sort the amount of data of different granularities from large to small. According to this order, we find the first corresponding granularity that is smaller than the maximum amount of data that can be displayed on the client. The granularity at this time can not only better represent the original time series trend but also avoid the situation in which the client data volume is too large. The granularity is jointly determined by the maximum amount of data that can be displayed by the client, the amount of data that the user queries, and the granularity in the historical data statistical table. The algorithm is illustrated in Algorithm 1.

Algorithm 1: Granularity selection algorithm. Algorithm 1: Select granularity (X, start time, end time, M)

Input: X time series ID, start and end time, client can display the maximum amount of data

FIGURE 7: Flow chart of trend query.

Output: < size of granularity >

- (1) Calculate the size of different granularity data of X Time Series in the historical statistics table from start time to end time;
- (2) Sort the amount of data in step 1 in descending order;
- (3) Select the first granularity that is smaller than the maximum amount of data that can be displayed on the client.

2.6.2. Inquiry Process. When the user provides a channel and time interval and then selects a statistic, if the user is not given, the median is the default statistic. The statistics currently supported are mean, median, maximum, minimum, and variance. The query process is illustrated in Figure 7. DisplayHistoryAction is the front-end Servlet, which accepts user requests and passes the request parameters to the background. HistoryQuery is the core processing class in the background and is responsible for interacting with the HistoryDataHandler class in the data layer. HistoryDataHandler encapsulates HBase's API for reading and writing data and is mainly responsible for interacting with HBase.

When the user is more interested in the data in a certain period of time, the user can select a period of interest in the front end to view more detailed trend information during this period.

2.7. Experimental Data. We use experimental results to illustrate the effectiveness of the distributed storage and calculation model DSCS-LTS designed in this study and the long-term series correlation coefficient calculation method

FIGURE 8: The effect of query length.

FIGURE 9: Effect of correlation coefficient threshold.

CCCA-LTS in a distributed environment. The specific test is based on the massive monitoring big data management platform, designed in this study. The main test indicators are divided into two parts: the influence of the sequence length change and the correlation coefficient threshold change of the monitoring time series on the query efficiency, and the experimental results are analyzed.

2.8. Data Sets. The experimental dataset comprises a large variety of sensor information collected from an industrial ketchup production line. The production line includes 100 sensors, such as thermometers, two-way/three-way accelerometers, and strain gauges. Each sensor corresponded to a time series. The sensor acquisition frequency was 1 Hz, that is, one acquisition value per second. We collected 5 years of data, which is about 250 GB.

2.9. Experimental Environment. The experimental cluster included five servers, and the operating system was Cen-

tOS8. Machine configuration memory to 64 GB, 8-core CPU, and 12 TB hard drive. The cluster includes 1 master node and 4 slave nodes. The Hadoop version was 2.6., and the HBase configuration was 1.0.

2.10. Experimental Result Analysis

2.10.1. The Query Efficiency Changes with the Query Length. The first experiments mainly tested the running time of CCCA-LTS and Bruteforce on the dataset and the number of algorithm iterations. The threshold was set to 0.7, and the specific test results are shown in Figure 8. Although the CCCA-LTS algorithm has more iterations, it is more efficient than the Bruteforce algorithm.

2.10.2. The Change of Query Efficiency with Correlation Coefficient Threshold. The second experiment mainly tests the efficiency of CCCA-LTS and Bruteforce when the threshold changes. The query length was 10^8 , and the

experimental results are shown in Figure 9. Because the Brute-force algorithm needs to take out all the data for calculation, the change in the threshold does not affect its operating efficiency, although the CCCA-LTS algorithm has many iterations, it does not need to read the entire sequence. It uses a segmented sequence to estimate the correlation coefficient of the entire sequence in an iterative manner, which is more efficient than the Brute-force algorithm.

3. Conclusion

Aiming at a massive monitoring data management platform, this study investigates the acquisition, storage, and analysis of big data monitoring. We designed a data collection terminal, collected sensor data to a distributed storage platform, through the NB-IoT network, and proposed a storage and calculation method called DSCS-LTS. According to the calculation of Pearson's correlation coefficient of a long time series, the algorithm CCCA-LTS is designed, which effectively improves the efficiency of the similarity query, by designing algorithms for particle size and query processes to solve the problem of visualization systems. In the development of the core platform to test the performance of the algorithm, the results show that the efficiency is better than that of the traditional method. Mass monitoring is an intelligent management reference for efficient time series data types.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no competing interests.

Acknowledgments

This work is supported by NSFC under grant 61962058, Integration of Industry and Education-Joint Laboratory of Data Engineering and Digital Mine (2019QX0035), and Bayingolin Mongolian Autonomous Prefecture Science and Technology Research Program (2021117).

References

- [1] "Influxdata [EB/OL]," 2021, <https://www.influxdata.com/products/influxdb/>.
- [2] W. Liu, T. Zhang, and J. Liu, "Window-based multiple continuous query algorithm for data streams," *Journal of Supercomputing*, vol. 75, no. 9, pp. 5782–5807, 2019.
- [3] "IoTDB [EB/OL]," 2021, <http://iotdb.incubator.apache.org/UserGuide/Master/IoTDB-Introduction/>.
- [4] "Parquet [EB/OL]," 2021, <http://parquet.apache.org/documentation/latest/>.
- [5] D. B. Lomet, "Data series management: the road to big sequence analytics," in *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms//International Conference on Foundations of Data Organization & Algorithms*, Chicago, Illinois, USA, 1993.
- [6] T. Palpanas, "Data series management," *ACM SIGMOD Record*, vol. 44, no. 2, pp. 47–52, 2015.
- [7] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM Transactions on Database Systems*, vol. 27, no. 2, pp. 188–228, 2002.
- [8] A. Camerra, T. Palpanas, J. Shieh, and E. Keogh, "iSAX 2.0: indexing and mining one billion time series," in *ICDM 2010, The 10th IEEE International Conference on Data Mining*, Sydney, Australia, 2010.
- [9] Y. Wang, P. Wang, J. Pei, W. Wang, and S. Huang, "A data-adaptive and dynamic segmentation index for whole matching on time series," *Proceedings of the VLDB Endowment*, vol. 6, no. 10, pp. 793–804, 2013.
- [10] K. Zoumpatianos, S. Idreos, and T. Palpanas, "Indexing for interactive exploration of big data series," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pp. 1555–1566, Snowbird Utah USA, 2014.
- [11] T. Rakthanmanon, B. Campana, A. Mueen et al., "Searching and mining trillions of time series subsequences under dynamic time warping," in *KDD '12: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, Beijing China, 2012.
- [12] M. Linardi and T. Palpanas, "Scalable, variable-length similarity search in data series," *Proceedings of the VLDB Endowment*, vol. 11, no. 13, pp. 2236–2248, 2018.
- [13] J. Peng, H. Wang, J. Li, and H. Gao, "Set-based similarity search for time series," in *SIGMOD '16: Proceedings of the 2016 International Conference on Management of Data*, San Francisco California USA, 2016.
- [14] Y. Liu, S. Hu, T. Rabl et al., "DGFIndex for smart grid," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1496–1507, 2014.
- [15] X. Lin, S. Ma, H. Zhang, T. Wo, and J. Huai, "One-pass error bounded trajectory simplification," *Proceedings of the VLDB Endowment*, vol. 10, no. 7, pp. 841–852, 2017.
- [16] J. Li, Z. Shen, and X. Meng, "Scientific big data management: concepts, technologies and system," *Journal of Computer Research and Development*, vol. 54, 2017.
- [17] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," in *International Conference on Data Engineering*, Minneapolis Minnesota USA, 2001.
- [18] E. Soroush, M. Balazinska, and D. L. Wang, "Array Store: a storage manager for complex parallel array processing," in *ACM Sigmod International Conference on Management of Data*, Athens Greece, 2011.
- [19] X. P. Yuan, "Estimation of three-way similarities based on connected bit minwise hash," *Journal of Shanghai Jiaotong University*, vol. 48, no. 7, pp. 936–941, 2014.
- [20] Y. Tao, K. Yi, C. Sheng, J. Pei, and F. Li, "Logging every footstep: quantile summaries for the entire history," in *ACM Sigmod International Conference on Management of Data*, Indianapolis Indiana USA, 2010.
- [21] W. Liu, T. Zhang, Y. Shen, and P. Wang, "Fast correlation coefficient estimation algorithm for HBase-based massive time series data," *Frontiers of Computer Science*, vol. 13, no. 4, pp. 864–878, 2019.