ISSN: 0974-5823 Vol. 7 No. 2 February, 2022

International Journal of Mechanical Engineering

Big Data-based large scale sensor data processing and analysis systems in CPS

[1]Dong-Jin Shin and [2]Jeong-Joon Kim

[1]Department of Computer Engineering, Anyang University, South Korea

[2]Department of ICT Convergence Engineering, Anyang University, South Korea

Abstract - The general aim of Cyber–Physical Systems (CPS), which links the virtual computing world and the physical real world with networks, is to create a control system that minimizes human intervention. This requires large scale sensor data collected through the sensor. However, conventional sensor data processing and analysis systems process all data required for analysis without additional processing, so although the accuracy of such analysis is high, the processing speed is slow. Therefore, this paper proposes a sensor data summarization technique that minimizes the accuracy degradation in the analysis and maximizes the sensor data processing speed. As a case study, we compare the processing speed and accuracy of the analysis using sensor data collected from plant automation.

Index Terms - Sensor, Cyber–Physical Systems, Big Data, Summary Technique.

INTRODUCTION

As the use of Big Data under the leadership of various advanced countries and the fourth industrial revolution for manufacturing innovation have been highlighted, various attempts have been made to actually incorporate related technologies. As a typical example, there is a smart factory that can apply Informational Communication Technology (ICT) technology to the manufacturing industry and analyze large-scale sensor data collected from facilities by changing the manufacturing paradigm from the mass production of small items to the small quantity production of various types of products. Constructing such a system requires a sensor network for data collection, a Cyber–Physical Systems (CPS) for modeling and simulating a system, autonomous control technology for controlling a large-scale Cyber–Physical Systems, a Hadoop Ecosystem, representative Big Data processing system for processing large scale data from the system, and industrial communication middleware technology that can unify the data formats of sensors manufactured by different manufacturers [1].

At the current level of technology, the purpose of a large scale sensor data collection system is active and efficient scheduling for effective resource operation. However, the ultimate goal is to introduce autonomous control to reduce the probability of system errors by minimizing human intervention. The most important research areas at present for realizing this goal are wireless sensor networks and the collection/processing/analysis of sensor data.

However, existing sensor data processing and analysis methods have several disadvantages: the data area commonly used in a large number of queries is repeatedly loaded from the disk, and all bulky data are received and analyzed when receiving the data, which results in high reliability but long execution time [2, 3]. Therefore, this paper proposes a system that applies data summary and memory sharing techniques in the process of the four stages of data processing—collecting, storing, processing, and analyzing—to solve the problems of existing research methods. We verified the proposed system by comparing and verifying the processing speed and accuracy according to whether summary and memory sharing techniques were used.

LITERATURE REVIEW

I. Industrial Communication Protocol

There are an increasing number of cases in which smart factories are being built using ICT technology for efficient resource management and effective process scheduling in factories. Establishing an integrated environment in the factory requires communication middleware that can manage facilities and sensors manufactured by different manufacturers. A typical example is OPC UA.

OPC UA is a standard used to control information exchange for sensor data and process procedures. It specifies the protocol that must be followed when the server that collects and manages sensor data has to communicate with the client that needs the data.

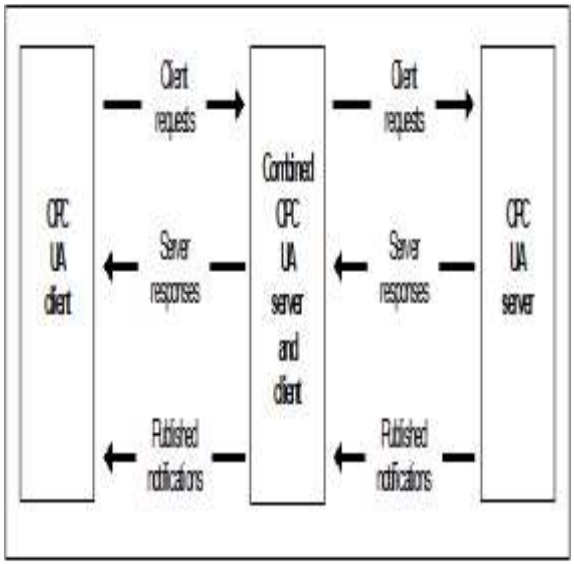

Figure 1 OPC UA System Architecture

Figure 1 shows the OPC UA system structure, which is a combination of server and client. Since the server and client use a data communication structure that is based on the transmission of request and response messages, the communication data is encoded in either XML or Custom Binary Format. This means that OPC UA is a platform-independent standard for communicating between various types of system, device, and sensor, each with different formats [4].

II. Technologies related to Big Data

The new concept called Big Data emerges as the amount of data increases and the number of types of unstructured data such as audio and video increases. Large-scale systems such as CPS and smart factories also generate large amounts of data, and many sensors collect them. Processing such large-scale sensor data requires collect–store–process–analyze steps such as the general big data processing steps [5].

There are many technologies for sensor data: Flume, Chukwa, SQOOP, Crawling for collection; HDFS, MongoDB, HBase, and ZooKeeper for storing; Hadoop, Pig, Hive, Spark, and MapReduce for processing; and R, Weka, and Mahout for analysis [6]. Figure 2 shows Among the techniques described above, we show the main architecture of spark used in this paper [7].

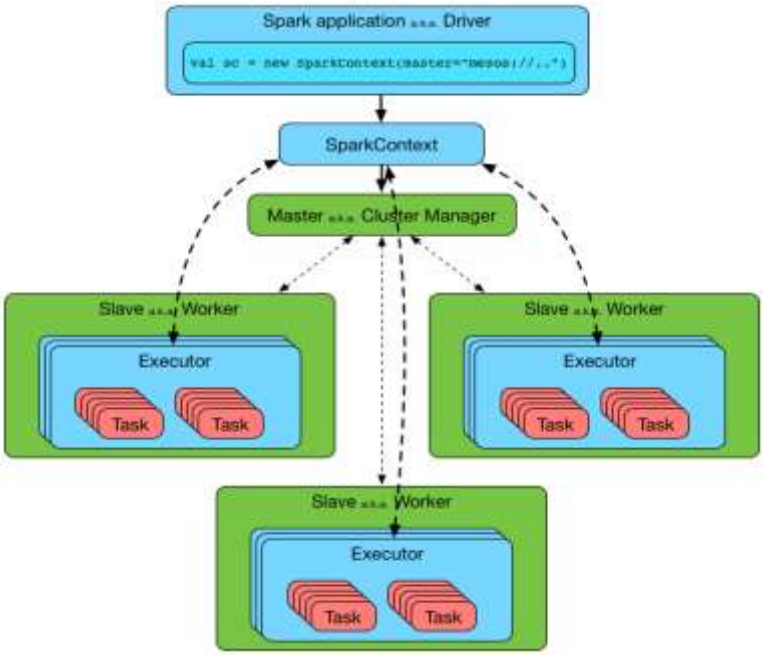

Figure 2 Spark Architecture

Apache Spark is an open-source cluster-computing framework. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. Apache Spark has as its architectural foundation the resilient distributed dataset (RDD), a read-only multiset of data items distributed over a cluster of machines, that is maintained in a fault-tolerant way. spark Core is the foundation of the overall project. It provides distributed task dispatching, scheduling, and basic I/O functionalities, exposed through an application programming interface (for Java, Python, Scala, and R) centered on the RDD abstraction. Spark SQL is a component on top of Spark Core that introduced a data abstraction called DataFrames, which provides support for structured and semi-structured data. Spark Streaming uses Spark Core's fast scheduling capability to perform streaming analytics. Spark MLlib is a distributed machine learning framework on top of Spark Core that, due in large part to the distributed memory-based Spark architecture, is as much as nine times as fast as the disk-based implementation used by Apache Mahout. GraphX is a distributed graph processing framework on top of Apache Spark [8, 9].

SUMMARY TECHNIQUES FOR SENSOR DATA

In the case of processing and analyzing large-capacity sensor data, all sensor data being generally accessible leads to a reduction of processing speed. There are data summary techniques that can improve this.

Data summary techniques that are proposed to solve the problem that the execution time increases when analyzing the whole dataset will summarize the data using an algorithm so that the system can perform analysis based on the summarized information. Typical algorithms used for summarizing data are histograms, wavelets, and wavelet histograms. The histogram method is a technique that divides all data into several sections and approximates the number of data items in the section to show the original data's distribution. Meanwhile, the wavelet method is a technique that represents the original data using wavelet coefficients obtained through wavelet transformation. The wavelet histogram is a combination of the histogram method and the wavelet method [10].

The histogram is one technique for effectively summarizing original data, and is widely used in selectivity estimation. Selectivity estimation means that the number of data items that belong to a certain range query is estimated. The histogram method is widely used in commercial database systems because it uses very little space and does not use information about the distribution of data to be summarized. The process of generating the Equi-width histogram, which is one histogram method, can be described as follows [11].

D={-7.7, -6.2, -8, 5, 3.2, 25, 22, 27, 31, 6.2, 9.8, 33, 33.2, 26, 20, 22, 19, 18.5, 23, 9.3, -6.2, -9.1}

When the above original data D exists, its domain is [-10, 34]. If the original data D is divided into B1–B5 and the bucket length is set to nine, the original data can be divided into five buckets from [-10, -1) to [25, 34). It is possible to convert the original data D into the form (bucket, item number) after counting the number of occurrences of items that belong to each bucket. In addition, the B1–B5 buckets represent five buckets. Figure 3 is a graphical representation of the histogram.

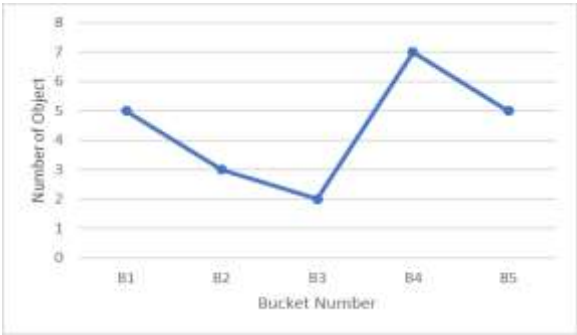

Figure 3 Graphical representation of Equi-width histogram

As shown in Figure 3, the items in original data D mainly belong to buckets B1, B4, and B5. Using the histogram in Figure 3, we can estimate the selectivity for the range query to output the number of data items belonging to [-2, 28].

First, the number of items of original data that belong to [-2, 28] is calculated in the following two types.

(1) Number of items of original data that belong to [-10, -1) and [-25, 34)

(2) Number of items of original data that belong to (-1, 25)

There is no bucket corresponding to [-10, -1] in the case of (1), but the number of items of original data that belong to [-2, -1] can be calculated as a percentage of the total number. Applying the following formula enables estimating the number of data items that belong to [-10, -2]

Estimated Value= (the endpoint value of the bucket-Starting point value of the estimated range)/(The endpoint value of the bucket-string point value of the bucket) (1)

When Equation (1) is applied, the estimated value of the number of items of original data that belong to [-10, -2] is ((-1)-(-2))/((-1)-(-10)) ×5=0.555. Likewise, the estimated value of the number of items of original data that belong to [28, 35] is ((25)-(28))/((34)-(25)) ×5=1.666. In the case of (2), the bucket corresponding to (-1, 25) is B2 = [-1,8], B3 = [8,17] and B4 = [17,25]. Therefore, the number of items of original data that belong to (-1, 25) is 3 + 2 + 7 = 12. When the calculation results of (1) and (2) are combined, the number of items of original data that belong to [-2, 28] is 0.555 + 1.666 + 12 = 14.222. The error between the summarized data and the original data was as small as 0.222 (∵|14 - 14.222| = 0.222). However, if the original data has characteristics that occur frequently for a certain data item, the error in estimating the number of data items may become large. For example, assuming that the number of items in bucket B5 is 100, the estimated value of the number of original data items that belong to [25, 28] changes to ((25)-(28))/((25)-(34)) ×100=33.3, and the number of items of original data that belong to [-2, 28] is 0.555 + 1.666 + 33.333 = 35.555. Then, the error between the summarized data and the original data is 21.555 (∵|14 - 35.555| = 21.555), which is very large compared to the former.

Wavelet is a typical data summary technique that has been used for query optimization and rough query processing in database systems. The wavelet transforms the original data into a wavelet coefficient through wavelet transform. Assuming that the same data as mentioned above is used, the wavelet method has a small error in summarizing the original data when the original data has a characteristic in which the specific data item frequently appears compared to the histogram method. The wavelet transform technique is based on the wavelet method; Haar Wavelet Transform (HWT) is the most widely used wavelet transform technique. The Haar Wavelet Transform converts the data to be summarized into Haar Wavelet coefficients. The Haar Wavelet coefficients consist of Average Coefficients and Detail Coefficients. The calculation process of the average coefficient and the detail coefficient is explained as follows [11]:

Arrangement of the number of items in dataset D:

V = {7, 5, 2, 8, 6, 7, 7, 8}

Assuming that there is one dataset D and v(v={v(1), … ,v(x)}) is an array of the number of items that appear in D, v can be regarded as a summary object and converted to wavelet coefficients through wavelet transform. Figure 4 shows the wavelet coefficient tree used to calculate the wavelet coefficients.

Figure 4 Haar wavelet coefficient tree

As shown in Figure 4, the Haar wavelet coefficient tree consists of nodes v(1)–v(8) and nodes w0–w7. The nodes v(1)–v(8) are the number of data items that v possesses. The nodes w0–w7 have the Haar wavelet coefficient, and the Haar wavelet coefficient consists of one average coefficient w0 and seven detailed coefficients w1–w7. The average coefficient w0 is the average value of v(1)–v(8), and the detailed coefficient is divided by Level 1–Level 3 and calculated in the order Level 3 → Level 2 → Level 1. The detailed coefficients can be calculated using Equation (2) below.

Coefficient= (∑▒〖(Value of Right Subtree)- 〗 ∑▒〖(Value of Left Subtree) 〗)/(The Number of Nodes in Subtree) (2)

The process of calculating the detailed coefficient w4 using Equation (2) is as follows. The nodes belonging to the left/right subtree of the node with w4 are v(1) and v(2), respectively. The sum of the right subtrees is five, the sum of the left subtrees is seven, and there are two nodes in the subtree. Therefore, the detailed coefficient w4 is -1(∴(5-7)/2 = -1). Another example is as follows; the nodes belonging to the left subtree of the node with the detail coefficient w2 are <{v(1), v(2)>, the nodes in the right subtree are <v(3), v(4)>, and there are four nodes in the subtree. When Equation (2) is applied, the detailed coefficient w2 is -0.5 (((2+8)-(7+5))/4=-0.5). The remaining detail coefficients can be calculated through Equation (2).

BIG DATA-BASED LARGE SCALE SENSOR DATA PROCESSING AND ANALYSIS SYSTEM IN CPS

I. System design and implementation

The real-time big data analysis system structure of manufacturing facility sensor data is composed of a factory that generates sensor data and a Collection/Storage/Process/Analysis Manager that collects, stores, processes and analyzes data generated by the factory and monitors to request and confirm information about collected data, as shown in Figure 5.

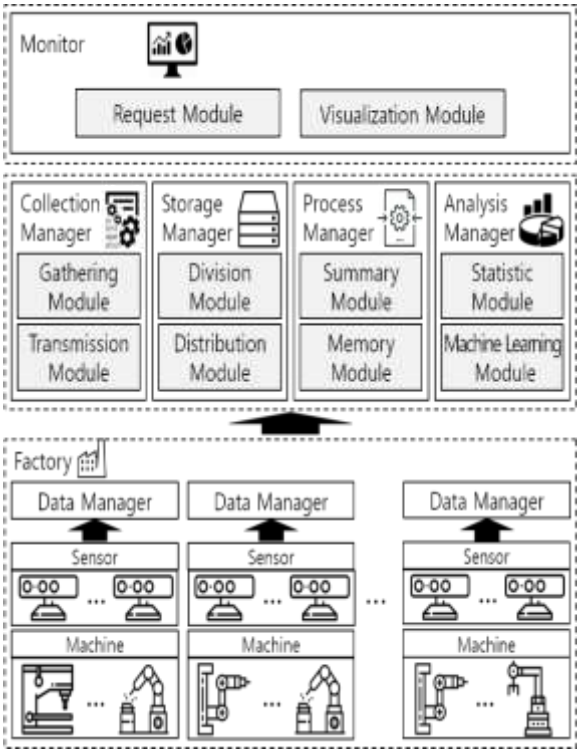

Figure 5 System Architecture

II. Factory

The factory consists of a machine, sensor, and Data Manager. The sensor detects the data generated by the machine and transmits it to the Data Manager. Since the machine may have been manufactured by different makers, the data format may differ. The Data Manager converts the different types of data collected from the sensors into a common format and transmits the data to the Collection Manager.

III. Collection Manager

The Collection Manager is an administrator that collects sensor data sent from the Data Manager; the Collection Manager consists of a Gathering Module that collects sensor data and a Transmission Module that transmits.

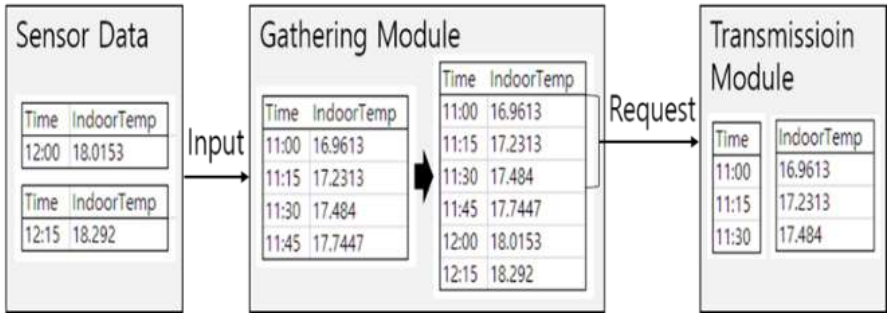

Figure 6 Design of Collection Manager

As shown in Figure 6, when the temperature data is input from the outside, the data is collected in a form to be added to the data that is already stored through the Gathering Module. When some of the stored data is requested from the Transmission Module, the Gathering Module returns the corresponding data. The returned data is divided and transmitted according to the column as shown in Figure 6, and is divided into Time and IndoorTemp in the Transmission Module.

IV. Storage Manager

The Storage Manager is an administrator that stores sensor data sent from the Collection Manager. The Storage Manager consists of a Division Module that divides the data sent from the Collection Manager into units of data and a Distribution Module that stores data.

Figure 7 The design of a Storage Manager

Figure 7 shows how data is stored in the Storage Manager through the Distribution Module. Day 1–3 data is stored in Slave 1–3 by the Master. If a problem occurs in the slave, data is lost; however, this is minimized by copying and storing the data for each day in different slaves. The master has metadata that records where each piece of data is stored in Slaves 1–3.

V. Process Manager

The Process Manager is a manager that processes data to analyze stored sensor data in the Storage Manager and consists of a Summary Module and Memory Module. The Summary Module reads, analyzes, and summarizes Big Data collected from sensors. Figure 8 shows an example table format of the sensor big data read.

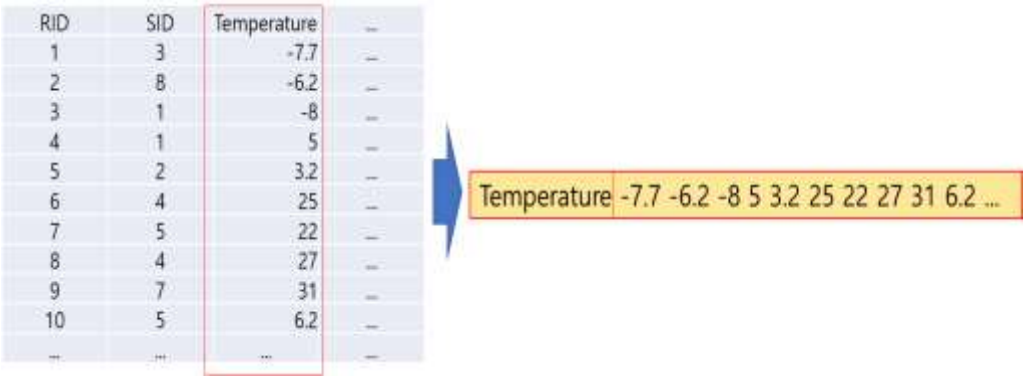

Figure 8 Example of extracting attribute values

As shown in Figure 8, the temperature attribute value is extracted from the attribute values of the data received from the data input module, and the parsed result becomes a one-dimensional data array.

In one map step, a local histogram is created with the temperature property value. The local histogram has the interval value of [-8, 32], so Key1–Key8 have the respective domains [-8, -3], [-3, 2], [2, 7], [7, 12], [12, 17], [17, 22], [22, 27], and [27, 32]. Since Key1, Key3, Key7, and Key8 respectively appear 3, 3, 2, and 2 times, their respective values are 3, 3, 2, and 2. Therefore, the generated local histogram is <(1, 3), (2, 3), (7, 2), (8, 2)>.

The histogram is composed of a pair of keys and values. Key is the only value that appears in the received attribute value, and the Value is the number of times the attribute value appears.

The wavelet histogram generation module receives and merges local histograms composed of (Key, Value) pairs from the histogram generation manager. In addition, the values of the same key are merged, and a global histogram is constructed by merging (Key, Value) pairs. Figure 9 is an example of merging the local histogram.

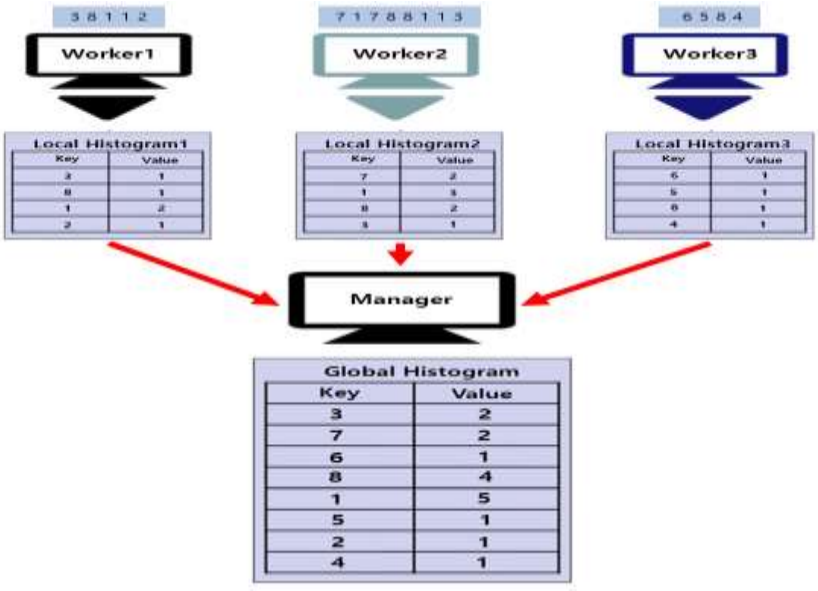

Figure 9 Example of local histogram

Figure 9 shows an example in which local histograms generated from each worker are generated as a global histogram through a reduce operation. Worker 1, 2, and 3 respectively generate local histograms 1, 2, and 3. Since local histograms 1 and 2 have Key1, Key3, and Key8 in common, (1, 2) and (1, 3) are merged into (1, 5). Similarly, the values of Key8 that are common to all workers are merged (8, 4). Merging all (Key, Value) generates a global histogram composed of <(3, 2), (7, 2), (6, 1), (1, 5), ...>. Finally,

the values of the (Key, Value) pair that constitute the global histogram are converted into wavelet histogram coefficients, generating a wavelet histogram composed of non-zero wavelet histogram coefficients.

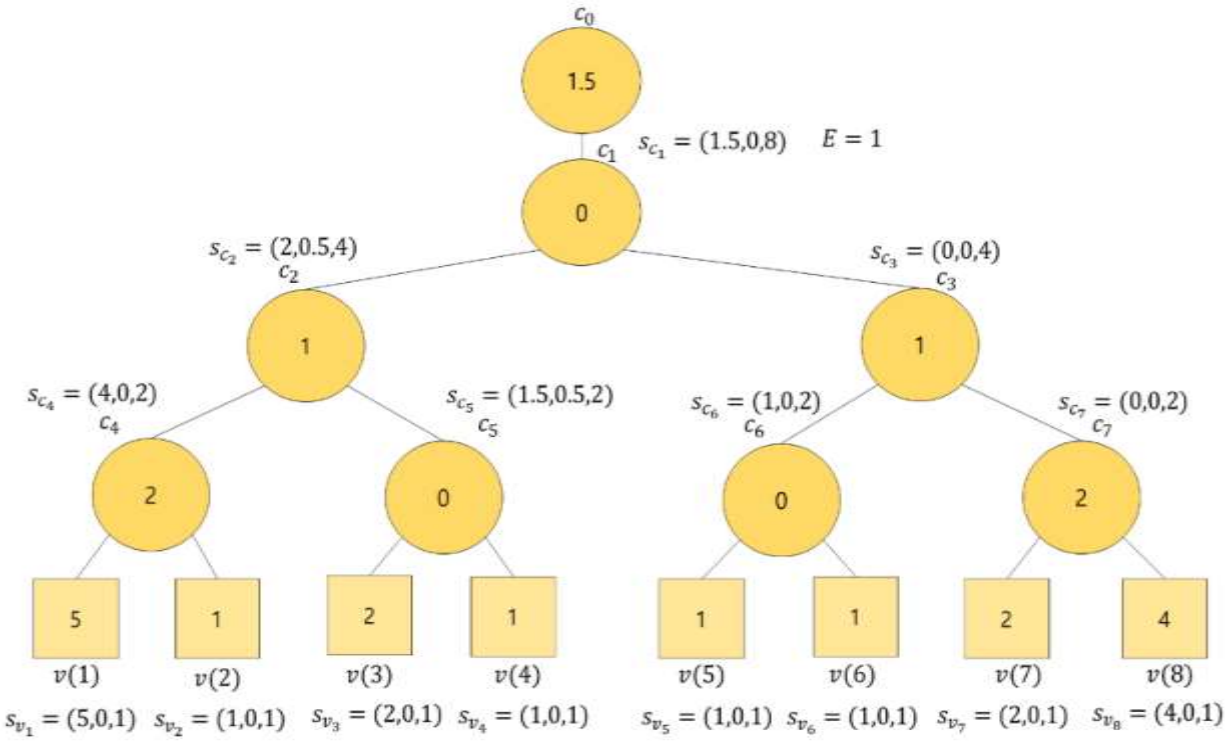

Figure 10 Example of local histogram

As shown in Figure 10, we start with node v1. Since the subtree of the v1 node is the v1 node itself, Sv1 for the leaf node such as v1 is calculated as (vi node value, 0, 1). Therefore, Sv1 = (5, 0, 1) and Sv2 = (1, 0, 1) for leaf nodes v1 and v2 are generated and added to S. In this case, since Sv1 = (al, dl, nl) = (5, 0, 1) for the left child node v1 of c2 and Sv2 = (ar, dr, nr) = (1, 0, 1) for the right child node v1 of c2 satisfying nl = nr = 1 exists in S, the wavelet histogram coefficient is calculated using Equation (3).

vmax and vmin for the c4 node are respectively 5 and 1, and is more than E( ), which means that d = 0, a = 4, n = 2, and c = 2. Thus, sc4 for c4 is added to S and sv1 and sv2 are removed from S. Continuing this process, if there is one s in S, calculate Equation (4) to calculate c0.

$$\frac{V_{max}-V_{min}}{2}\begin{cases} \geq E,\ c = (a_l - a_r)/2,\ d = \max\{d_l, d_r\},\ a = a_l\text{-}c,\ n = 2n \\ < E,\ c = 0, d = (V_{max} - V_{min})/2,\ a = (V_{max} + V_{min})/2,\ n = 2 \end{cases} \quad (3)$$

IF $|a| \geq |E-d|$,THEN c=a (4)

The sc0 = (1.5, 0, 8) remaining in S according to Equation (4) is c0 = 1.5 because $|1.5| \geq |1 - 0|$. Finally, non-zero c0 and c3 are stored as final wavelet histogram coefficients.

The Memory Module is a module for memory use and memory sharing technology to reduce memory usage and the query processing time when a large number of queries are received for sensor data processing. Memory sharing technology is a technique that shares common data among a large number of queries in a memory area, unlike a general processing method that allocates a memory area for each query and processes the query. This eliminates the need to allocate memory areas for each query, which allows efficient memory space usage and reduces the amount of work required to load data from the disk, thereby reducing execution time. Figure 11 shows the memory sharing process [13, 14].

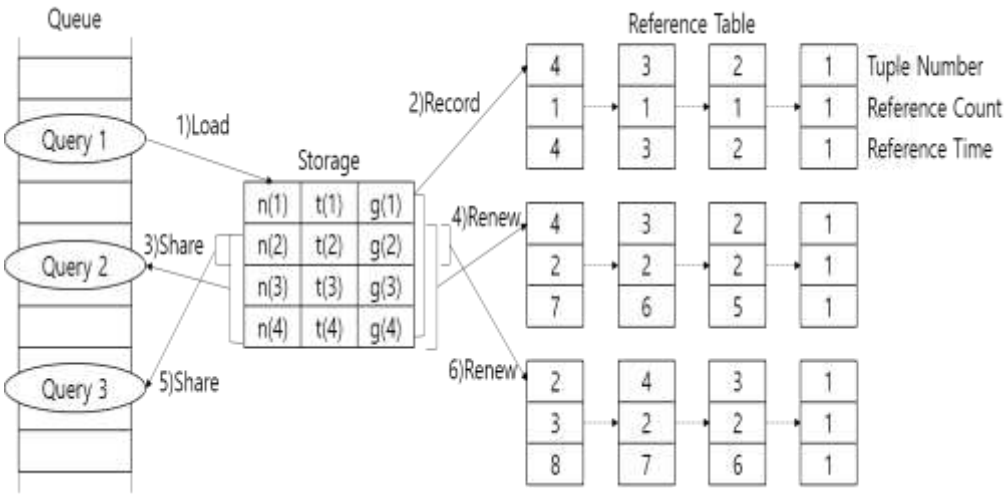

Figure 11 Design of Memory Module

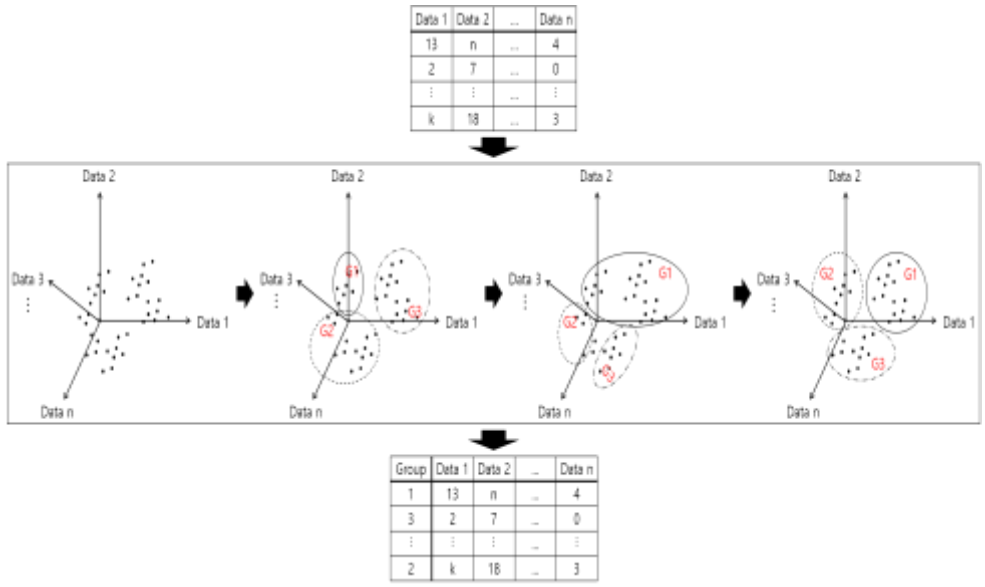

Figure 12 Design of Analysis Manager

Query 1 fetches data corresponding to Tuple Number 1–4. The information is recorded in the Reference Table, which records the Tuple Number, Reference Count, and Reference Time. Next, since Query 2 requested data corresponding to Tuple Number 2–4, it can be confirmed that the reference count and reference time information corresponding to Tuple Number 2–4 of the Reference Table have been updated. Finally, Query 3 requests data corresponding to Tuple Number 2, and the reference count and reference time information corresponding to Tuple Number 2 in the Reference Table are updated. If there is information that is not continuously used, such as Tuple Number 1, that information should be deleted so that memory space can be reserved for future queries.

VI. Analysis Manager

Analysis Manager is a manager that analyzes data stored in the Storage Manager or processed data through Process Manager. It is composed of Statistic Module and Machine Learning Module. The Statistic Module uses refined data from the Process Manager and computes from elementary statistics such as the mean, standard deviation, and error rate to advanced analytical results such as predictions, classifications, and clusters. The Machine Learning Module performs analysis for artificial intelligence and machine learning such as reinforcement learning and deep learning using both statistical module analysis and refined data.

Figure 12 shows the process of cluster analysis among several analyses. When data of n categories such as factory inside/outside temperature, humidity, CO2 concentration, and sunshine amount are inputted, it is expressed in n-dimensional space and data is grouped using the Euclidean Distance (√(∑▒(p_i- q_i)^2)).

The Analysis Manager also needs analysis to interpret the information summarized in the Process Manager. Equation (5) below is a formula for restoring the wavelet to analyze the wavelet coefficients. D(i) is the restored value of v(i) and wj is the node value of the wavelet coefficient. Sign(i, j) is used as a symbol that represents the inequality of the wj value, and path(i) refers to a route from the leaf node to the root node. Starting from the root node, v(i) can be expressed as sign(i, j) = 1 on the left subtree surface of wj, and sign(i, j) = -1 on the right side, and the final restoration value can be expressed as D(i)=∑_(j∈path(i))▒〖Sign(i,j)× w_j 〗.

$$D(i) = \sum_{j \in path(i)} Sign(i,j) \times w_j$$
$$Sign(i,j) = \begin{cases} i,j = 0 \text{ or } v(i) \in left\ subtree\ of\ w_j \\ -1, v(i) \in right\ subtree\ of\ w_j \end{cases} \quad (5)$$

Restoring the v2 node value of Equation (5), requires that D(2) is computed by performing an operation between cj that belongs to path (2). Therefore, the cj belonging to path(2), which is the path from v2 to c0, are <c0, c1, c2, c4>. If j = 0, sign(2, 0) × c0 = 1.5. sign(2,1) × c1 = 0, sign(2,2) × c2 = 1, and sign(2,4) × c4 = -2, because the v2 node belongs to the left subtree of the c1 and c2 nodes and the right subtree of the c4 node. Therefore, D(2) = 1.5 + 0 + 1 - 2 = 0.5. In this manner, the restoration values of nodes v1–v8 are <4.5, 0.5, 2.5, 2.5, 1.5, 1.5, 1.5, 3.5>. Since the error calculated by the L∞ calculation method is <0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5>, it does not exceed the specified error boundary (E = 1).

VII. Monitor

The Monitor is responsible for requesting data from the Analysis Manager and visualizing received data. When the Analysis Manager sends relevant data, the Visualization Module uses the transferred data to represent it as an appropriate graph, chart, or table.

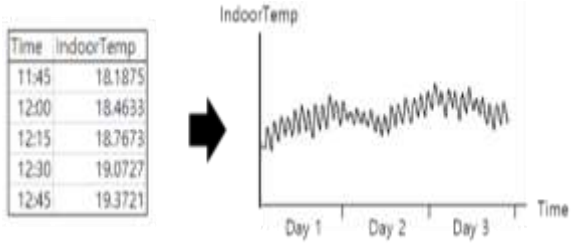

Figure 13 Design of Monitor

Figure 13 shows an example of visualization in which the time and temperature data are input to create a time series graph. The visualization module can be expressed in various ways such as graphs, tables, charts, etc. in accordance with predefined data conditions.

PERFORMANCE EVALUATION

The proposed system's key point is its minimization of the analysis accuracy degradation and its improvement of the processing time for queries compared to the existing system through both the summary techniques and the memory sharing technique. Therefore, this paper measures the accuracy and processing speed of the analysis, and the memory usage and memory performance to summarize the sensor data.

The system has 16 clusters in total, and Driver of Spark are applied to one system. Spark's workers were applied in the remaining 15 units. The Driver of Spark has an Inteli5 CPU and 4 GB of RAM. The remaining 15 worker have Inteli5 CPUs and 2 GB of RAM.

I. Summary of sensor data

This paper analyzed sensor data information through data summary techniques, so there are possible errors. However, it can be predicted that the processing speed will be improved. This paper compares and analyzes the accuracy and processing speed of Full Scan, Range, Aggregate and Join queries for 1,000–500,000 ten thousand data inputs. Figure 14 below shows the accuracy.

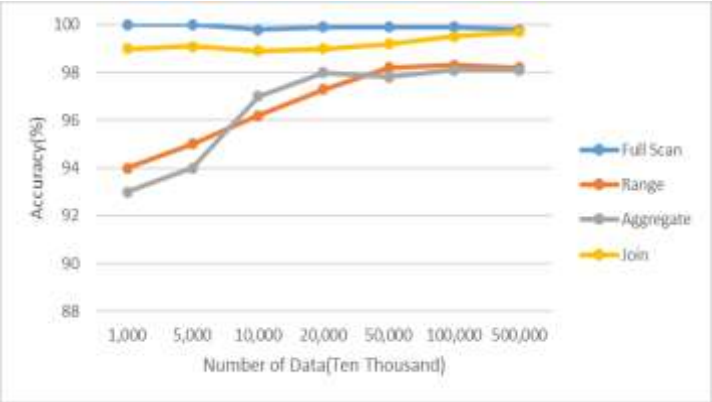

Figure 14 Accuracy of analysis

As shown in Figure 14, the full scan shows close to 100% accuracy with some errors. This shows high accuracy in direct access to all data. The Range and Aggregate queries can be approximated by examining a summary wavelet histogram without having to directly access the disk. However, there is a problem that cannot be tolerated. Figure 15 below shows the query processing speed for the summary technique.

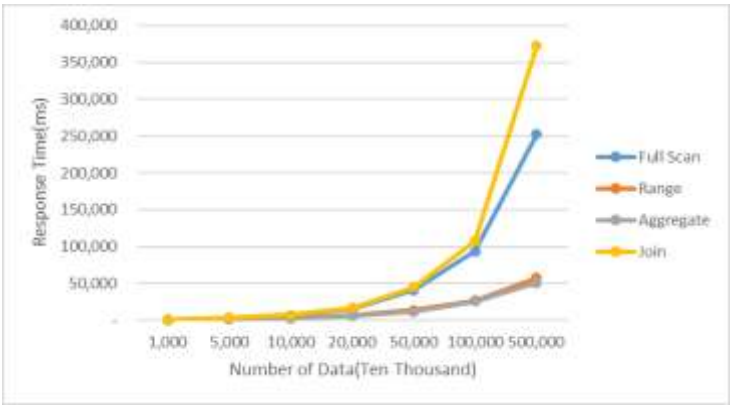

Figure 15 Processing Speed

As Figure 15 shows, the aggregate query and range query show very fast responses because the histogram wavelet can be reconstructed without physically accessing the disk, and thus results can be obtained quickly. However, it also shows that the speed increase is not prominent for join operations and full table scans that can only access the disk.

II. Memory Sharing

The performance of memory sharing is measured to demonstrate memory usage and query processing through the memory sharing technique that this paper proposes. This paper measures memory usage and query processing time with 1, 5, and 10 thousand queries both when memory sharing was used and when it was not.

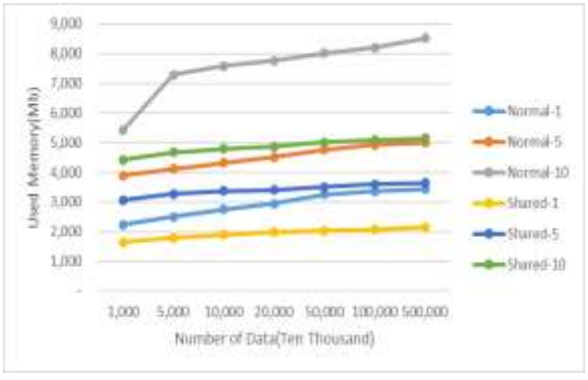

Figure 16 Usage of memory

As shown in Figure 16, memory space due to the use of memory sharing becomes more efficient as the number of queries and data to be processed increases.

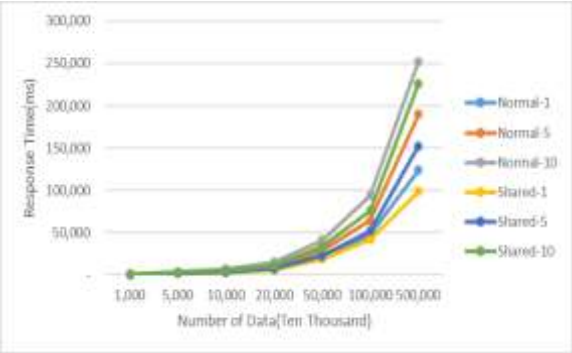

Figure 17 Memory performance

As shown in Figure 17, the memory sharing technique can improve the query speed to some extent. However, it can also be seen that the performance improvement slows down as more data and more queries are generated. Memory sharing technology cannot expect a performance increase due to frequent reference table updates and beta memory locks if there is a large amount of data and many queries.

CONCLUSION AND FUTURE WORK

This paper proposes a processing and analysis system for large capacity sensor data. The existing system has the disadvantage of high accuracy of analysis but slow processing speed because it accesses all data necessary for sensor data processing and analysis. In addition, processing many queries can create a heavy memory usage load because the resources used in common are not considered. Therefore, this paper proposes sensor data summary and memory sharing techniques to solve these problems. In the proposed system, it is confirmed that the analysis accuracy with the introduction of the summary technique is < 10%, but the processing speed can be improved through self-performance evaluation. The memory usage can be reduced by introducing memory sharing technology, and the performance improvement has been confirmed. However, the amount of shared memory increases in proportion to the amount of data, as shown in the experiment evaluation in Section 4. Therefore, if the amount of data exceeds a certain level, the performance improvement due to memory sharing is insufficient. Therefore, future studies should seek to resolve this problem.

ACKNOWLEDGMENT

None.

CONFLICTS OF INTEREST

the authors have no conflicts of interest to declare.

REFERENCES

[1] Wang S, Zhang C, Liu C, Li D, Tang H. Cloud-assisted interaction and negotiation of industrial robots for the smart factory. Computers & Electrical Engineering. 2017; 63:66-78.

[2] Andreou P, Zeinalipour-Yazti D, Pamboris A, Samaras G. Optimized Query Routing Trees for Wireless Sensor Networks. Journal of Information Systems. 2011; 36(2):267-291.

[3] Taslidere E, Cohen F S, Reisman F K. Wireless Sensor Networks-A Hands-On Modular Experiments Platform for Enhanced Pedagogical Learning. Journal of IEEE Transactions on Education. 2011; 54(1):24-33.

[4] Gruner S, Pformmer J, Palm F. RESTful Industrial Communication With OPC UA. IEEE Transactions on Industrial Informatics. 2016; 12(5): 1832-1841.

[5] Lei P, Zheng L, Wang L, Wang Y, Li C, Li X. MTConnect compliant monitoring for finishing assembly interfaces of large-scale components: A vertical tail section application. Journal of Manufacturing Systems. 2017; 45:121-134.

[6] Oussous A, Benjelloun F Z, Lahcen A A, Belfkih S. Big Data Technologies: A Survey. Journal of King Saud University, 2018; 30(4):431-448.

[7] Apache Spark 3.2.0. https://spark.apache.org/docs/latest/. Accessed 5 January 2022.

[8] Saliha M, Ali B, Rachid S. Towards large-scale face-based race classification on spark framework. Multimedia Tools and Applications. 2019; 78(18): 26729-26746.

[9] Mera D, batko M, Zezula P. Speeding up the multimedia feature extraction: a comparative study on the big data approach. Multimedia Tools and Applications. 2017; 76(5):7497-7517.

[10] Ma Z, Liu W. Outlier correction method of telemetry data based on wavelet transformation and Wright criterion. Multimedia Tools and Applications. 2016; 75(22): 14477-14489.

[11] Garafalakis M, Gibbons P B. Probabilistic Wavelet Synopses. Journal of ACM Transactionson Database Systems. 2004; 29(1): 43-90.

[12] Pang C Y, Zhang Q, Zhou X, Hanse D, Wang S, Maeder A. Computing Unrestricted Synopses under Maximum Error Bound. Journal of Algorithmica. 2013; 65(1):1-42.

[13] Krizanovic K, Galic Z, Baranovic M. Spatio-Temporal Data Streams: An Approach to Managing Moving Objects. In Proceedings of the International Convention 2010 (pp. 744-749) MIPRO.

[14] Park J, Kim K, Ahn S, Hong B. Continuous Query Processing on Data Stream: Sensor, Location and Identification. In Proceedings of 7th International Conference on Information Technology: New Generations 2010 (pp. 518-522). IEEE.