2018 IEEE 32nd International Conference on Advanced Information Networking and Applications

Fog Computing with Distributed Database

Tsukasa Kudo
Faculty of Informatics
Shizuoka Institute of Science and Technology
Fukuroi, Japan
kudo.tsukasa@sist.ac.jp

***Abstract*—In recent years, with the progress of IoT, the entry data from various sensors is accumulated on the cloud server and used for various analyses as big data. On the other hand, in order to transfer a large amount of data to the cloud server, there were the problems such as the restriction of network bandwidth, and the delay of feedback control of the sensors. For these problems, Fog computing has been proposed in which the primary processing of the sensor data is performed at the fog node installed near the sensors, and only its processing results are transferred to the cloud server. However, in this method, in the case where the original data of the sensor is required for various analyses at the cloud server, such a data must be transferred additionally. That is, a mechanism is necessary to manage the data of the entire system and to mutually utilize it. In this paper, I propose a data model which consists of three levels: the first level saves the original sensor data and is placed in the fog node; the second level saves the extraction data extracted by the primary processing; the third level saves the analysis results data. The second and third levels are placed in the cloud server. And, by constructing this data model with a distributed database, it can be performed efficiently to refer the arbitrary original sensor data in the fog nodes from the cloud server. Moreover, I implement this reference processing in two ways using MongoDB, which is a kind of NoSQL database, to evaluate this data model. And, I show it is necessary to select the reference way according to the system environment: the network bandwidth, the database performance of the fog node and cloud server, and the number of the fog nodes.**

***Index Terms*—IoT, Fog computing, distributed database, NoSQL database, MongoDB, data model**

I. Introduction

In recent years, with the progress of IoT (Internet of Things) [1], utilization of sensor data is spreading in various fields. Furthermore, it became to be possible to accumulate and share these data in the cloud server and to use for various analyses as big data. On the other hand, since a large amount of data such as video is transferred from a large number of sensors to the cloud server, it is pointed out that there are the problems such as the restriction of network bandwidth and the delay of feedback control of the sensors.

For these problems, Fog computing [2] (or Edge computing [3]) have been proposed. Here, the conventional cloud computing transfers all the sensor data to the cloud server. On the other hand, as for Fog computing, the sensor data is primarily processed in the fog node, which is installed closer to the sensors, and only its results are transferred to the cloud server. For example, in many buildings of enterprises and schools, surveillance cameras are installed at such as the entrances and areas where security should be maintained. And, they are used to detect such as the invasion of suspicious individuals, and occurrence of the abnormalities. Here, as for the cloud computing, since almost only the background is shot at a time when the traffic is low, only redundant data is transferred to the cloud server. On the other hand, by utilizing Fog computing, the following efficient monitoring can be performed: the data is transferred to the cloud server only while the sensor detects the movements of objects; the sensor can track the movement of Object rapidly.

However, in the case where some incident occurs, it becomes to be necessary to analyze the sensor data in detail to clarify the background. That is, other data has to be transferred to the cloud server: the data of the time period before and after the incident, the data of related sensors, and so on. This means that the following mechanism is necessary: all the sensor data should be saved in the fog node; the data in fog node should be referred by the cloud server when it is necessary to analyze the background.

Here, considering the rapid increase of the sensors, it is expected that the fog node will also increase rapidly. So, it is necessary to access the data of these many fog nodes efficiently and securely from the cloud server. However, it is pointed out that there are the problems with the data management, security and backup in the case of implementing such a mechanism only by using the file system [4].

On the other hand, in recent years, various NoSQL databases have been proposed and put to practical use in order to access such a diverse and large volume data efficiently [5]. Since these databases were constructed on the premise of a distributed environment, it is possible to be utilized even for the widely deployed fog nodes. In particular, MongoDB provides GridFS interface to manipulate an enormous data, which is a kind of NoSQL database [6]. And, it has been shown that it can manipulate the enormous data more efficiently than the traditional relational database [7].

In this paper, I propose a data model which consists of three levels: the first level saves the original sensor data and is placed in the fog node; the second level saves the data extracted by the primary processing; the third level saves the analysis results data. The second and third levels are placed in the cloud server. And, constructing this data model by using the distributed database, it is possible to refer the necessary original data of the first level in the fog node from the cloud server. For example, as for the above-mentioned surveillance camera, the

1550-445X/18/$31.00 ©2018 IEEE
DOI 10.1109/AINA.2018.00096

data transferred to the cloud server in Fog computing is saved into the second level; additionally, the original video data is saved into the first level. Then, when the additional original data is necessary for the analysis in the cloud server, the data of the first level in the fog node can be referred from the cloud server.

Moreover, I implement this reference feature in two ways to perform the comparative evaluations between them: the first way is the server-side reference, in which the original data in the fog node is referred directly from the cloud server by the data manipulation feature of the database; the second way is the node-side reference, in which the data is extracted from the original data at the fog node, then transferred to the cloud server to save into the second level. And, I show the important requirement to choose one of these ways: the network bandwidth, database performance on the fog node and cloud server, and the number of the fog nodes.

The remainder of this paper is organized as follows. Section II shows the related works and my goal of this study, and I propose the data model for Fog computing based on the distributed database in Section III. Section IV shows the implementation case and its evaluation results, and Section V shows the discussions on these results. Lastly, I conclude this paper in Section VI.

II. RELATED WORKS AND GOAL OF THIS STUDY

Along with the progress of IoT, it is expected that major information of big data will be generated by analyzing the data entered from various sensors [8]. In addition, with the widespread use of wearable sensors, it is expected that these sensors are used not only in various business fields but also in various places such as the medical site and factories [9]–[11].

However, in order to process such a large amount of sensor data in the cloud server, it was pointed out that there are the problems on the network such as the restriction of network bandwidth and the delay of feedback control of the sensors. So, Fog computing has been proposed, in which the primary process of the sensor and the feedback control of the sensors are performed in the fog node installed closer to the sensors [2]. In addition, as for Fog computing, several architectures have been shown: its reference model [12], Dew computing in which user devices can be used even in the ad hoc network environments [13], and so on.

On the other hand, as for the database, to treat efficiently the various and enormous data, such as the sensor data, various NoSQL databases have been proposed and put to practical use [5]. Since these are premised on the distributed database environment, it is expected that its data manipulation can be performed efficiently even in Fog computing environment.

Among these, MongoDB is classified in the document-oriented database which data structure is expressed by JSON [14]. That is, its structure is according to the tree structure as shown in Fig 1. Here, collection “list” consists of documents “member”. The former corresponds to a table of RDB (relational database), and the latter corresponds to records. And, since each attribute is expressed by the pair of the attribute name and its value, each document is able to have the individual structure. That is, since MongoDB does not have a scheme of the collection unlike RDB, it can save data with various structure into the single collection. Moreover, it provides GridFS interface to treat the enormous data [14], [15]. In GridFS, enormous binary data such as the pictures and videos is saved into two collections: one collection saves the binary data and the other saves its metadata. This metadata structure is expressed by JSON shown in Fig. 1. And, arbitrary data can be added as the metadata according to business needs; the target binary data can be searched by using the metadata as a search key. That is, the binary data can be managed by using the metadata. And, it was shown that it had better performance than the relational database as for the simple CRUD operations and enormous data manipulation [7], [16].

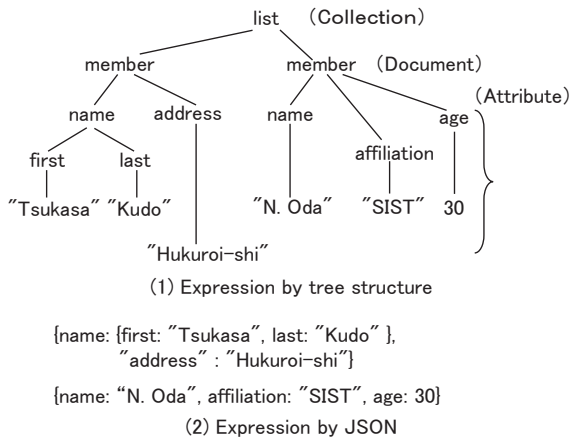

Fig. 1. Data structure of MongoDB.

Incidentally, MongoDB has some restriction compared with the relational database, such as the join operation of the collections and the transaction processing across the multiple data. As for the former, such an operation is not provided; as for the latter, the complete ACID properties are not maintained on the plural document manipulation, which is consist of the Atomicity, Consistency, Isolation, and Durability. However, since the target of this study is the simple manipulation of the sensor data, it does not need such a complex feature. Furthermore, as for Fog computing, it is expected that a large number of fog nodes are spread in the wide area, and process the entered sensor data. However, it has been pointed out that in the case of managing an enormous data of such a node by the file system, there are problems such as data management, security, and backup [4].

For the above reasons, by storing the original sensor data in the fog node by using MongoDB, following effects is expected. That is since the original sensor data can be referred from the cloud server efficiently as necessary, the issue of Fog computing that the necessary data is missed for the analyses in the cloud server is solved. In addition, it improves the complexity of the data and security management in the fog

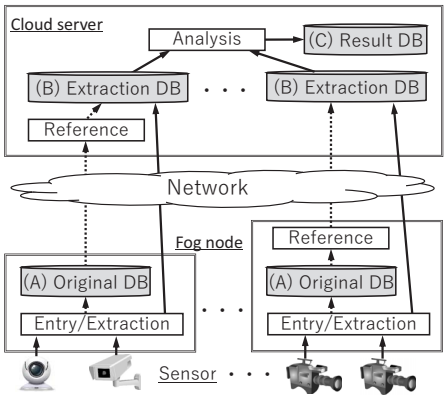

Fig. 2. Construction of data model for Fog computing.

node. However, I could not find the proposal to apply the distributed database to the fog nodes in order to compensate the necessary data for the analysis in the cloud server.

Therefore, in this paper, I propose a distributed data model consisting of three levels: the original sensor data is placed in the fog node; the extracted data and analysis result data are placed in the cloud server. And, this data model assumes the distributed database. In this data model, the primary process results of fog node are saved as the extracted data in the cloud server for the analysis process; the original data can be referred by the cloud server if necessary. So, my goal of this study is to show the performance of this model composed by using MongoDB, and especially to show the effective composition of the reference feature of this model.

III. DATA MODEL FOR FOG COMPUTING

A. Construction of Proposal Data Model

In Fog computing, the original data is entered from the sensors to the fog nodes, and the primary processed results are saved into the cloud server. And, in the cloud server, these results are analyzed and its results are saved as the analyzed results. So, in this paper, I propose the data model consisting of three levels as shown in Fig. 2, in which Fog computing is modeled from the viewpoint of data accumulation.

The first level is "Original database" (shown by "Original DB" in Fig. 2, hereinafter, are same) shown by (A) in Fig. 2, which saves the original data entered from sensors. Similarly, the second level is "Extraction database" shown by (B), which saves the primary processed results, that is, the extracted data from the original data. The third level is "Result database" shown by (C), which saves the results of the analysis in the cloud server.

In Fig. 2, the solid arrows show the data flow of the entry sensor data as for conventional Fog computing. That is, only the primary processed results are transferred to the cloud server and saved in (B). On the other hand, the broken arrows show the data flow to refer the original sensor data, which is proposed in this paper. That is since each level of this data model is constructed by the distributed database, it becomes possible to access mutually between the fog nodes and cloud server.

For example, as for the case study of the surveillance camera mentioned in Section II, only the part of data, where the movement of the object is detected, is extracted at the fog node and transferred to the cloud server to save into (B). The cloud server analyzes this data to create the results such as the number and time of passers and saves the results into (C) to share them.

In addition, in this data model, the original data of the surveillance camera is saved into (A) concurrently with saving the primary process result into (B). And, it can be referred from the cloud server. So, in the case where an incident occurred, cloud server can refer additionally the data of the target time period and other related cameras to analyze this incident. That is, necessary data is saved into (B), then the incident is analyzed.

B. Performance of Proposal Data Model

As for the proposed data model, the operation to save the original data is added. So, I show the performance of this data model from the viewpoint of data entry and reference. In addition, as for the data reference, I show the following two ways: as for the first way, the server-side reference, the cloud server refers to the Original database directly, then the data is extracted to save into the Extraction database; as for the second, the node-side reference, the original data reference is performed in the fog node, and only the extracted data is transferred to the cloud server to save into the Extraction database.

In this data model, the elapsed time of data entry from the sensors to fog node of number i is expressed by equation (1).

$$E_i = \frac{c_i}{W_i} + \frac{c_i \alpha_i}{\hat{W}} \tag{1}$$

Here, α_i is the extraction ratio from the original data, which is transferred to the cloud server as the extraction data; $\hat{W}$ is the performance to write from the fog node to the database of cloud server via the network; W_i is the performance that the fog node number i writes the own database. And, c_i is the entry data amount of the fog node number i per unit time expressed by equation (2).

$$c_i = \sum_{j=1}^{m_i} c_{ij} \tag{2}$$

Here, as for the fog node number i, c_{ij} is the entry data amount per unit time from the sensor of number j; m_i is the number of sensors of this fog node. In Equation (1), the first term is the elapsed time to transfer the extracted data to the cloud server, which is the operation performed in the conventional Fog computing, too. So, to compose the proposal data model, the elapsed time expressed by the second term increases in the data entry from the sensors.

The elapsed time E of data entry as for the whole fog node is expressed by using function g as shown in Equation (3).

$$E = g(\frac{c_i}{W_i} + \frac{c_i \alpha_i}{\hat{W}}) \tag{3}$$

Here, the first term of Equation (3) shows the elapsed time that each fog node writes into its own Original database. So, they are performed concurrently. Also, the second term shows the elapsed time that all the fog nodes write into the Extraction database via the network. So, though they involve delays caused by network bandwidth and database access, they are performed concurrently. That is, the time of function $g(x_i + y_i)$ is within the range of time expressed by Equation (4).

$$max(x_i + y_i) \leq g(x_i + y_i) \leq \max(x_i) + \sum_{i=1}^{n} y_i \quad (4)$$

That is, E is the time obtained by adding the following at most: first, the largest elapsed time of each fog node to write into the own Original database; second, the total time to write the data extracted at each fog node into the Extraction database via the network.

Next, as above-mentioned, there are two ways to refer the fog node data from the cloud server. The first, the server-side reference, is to refer the Original database of the fog node directly from the cloud server, then the required data for the analysis is extracted from this reference data to save into the Extraction database. This corresponds to the fog node on the left side in Fig. 2. This elapsed time S is expressed by Equation (5).

$$S = \sum_{i=1}^{n} (\frac{d_i}{\hat{R}_i} + \frac{d_i \beta_i}{W}) \quad (5)$$

Here, d_i is the amount of data to be read from the Original database; β_i is the extraction ratio of the original data, which is extracted from this read data to save into theExtraction database; $\hat{R}_i$ is the performance to read from the database of fog node number i via the network; W is the performance to write into the own database at the cloud server; n is the number of target fog nodes. For example, in the case where the incident occurs for the surveillance camera as shown in Section III-A, the related data with it in Original data must be referred. It is referred in this way and saved into the Extraction database.

The second, the node-side reference, is performed in each fog node: the own Original database is referred, then the necessary data is extracted to save into the Extraction database. In this case, the former is performed concurrently at each fog node, and latter is same as to write into the Extraction database as for the data entry. That is, this elapsed time is expressed by Equation (6) including above-mentioned function g, similar to Equation (3).

$$F = g(\frac{d_i}{R_i} + \frac{d_i \beta_i}{\hat{W}}) \quad (6)$$

That is, F is the time obtained by the sum of the following at most: first, the largest elapsed time to refer the own Original database at each fog node; second, the total time to write the extracted data at each fog node into the Extraction database via the network.

I show a simple case having single fog node. Here, let the elapsed time to write is twice the time to read; let the elapsed time to read and write via the network is twice the time as of to read and write the database in own fog node or cloud server. That is, this is the case shown by Equation (7).

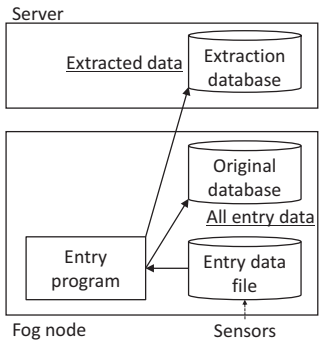

Fig. 3. Implementation of sensor data entry function.

$$R_i = 2\hat{R}_i = 2W = 4\hat{W} \quad (7)$$

In this case, S and F become as follows.

$$S = \frac{2d_i}{R_i} + \frac{2d_i \beta_i}{R_i} = \frac{d_i(2 + 2\beta_i)}{R_i} \quad (8)$$

$$F = \frac{d_i}{R_i} + \frac{4d_i \beta_i}{R_i} = \frac{d_i(1 + 4\beta_i)}{R_i} \quad (9)$$

So, if $\beta_i = 0.5$ then the both are equal to $3d_i/R_i$, which shows the extraction ratio to be transferred of the read data is 50%. And, if β_i is less than 0.5, then F is shorter than S; if β_i exceeds 0.5 then S is shorter than F. For example, if $\beta = 0.25$ then $S = 2.25d_i/R_i$ and $F = 2d_i/R_i$. That is, the elapsed time of reference performed in the fog node is 12.5% faster than the time performed in cloud server.

IV. EXPERIMENTS AND EVALUATIONS

To evaluate the performance of the proposal data model by the actual system, I implemented the sensor data entry feature and the two Original data reference features as a prototype, which is mentioned in Section III-B. As the environment of this experiment, I used two PCs connected via 1 Gbit HUB as the fog node and server. The fog node's PC equipped i7-7700CUP 3.6 GHz, 32 GB memory, 512 GB SSD; the server's PC equipped i7-6700 CPU 3.4 GHz, 16GB memory, 480 GB SSD; and, both operating systems were Windows 10. In addition, as for the both, I used MongoDB 3.4.9 as the database; the program was built in Java SE-1.8 and MongoDB Java driver 3.5.0. Incidentally, the MongoDB Java driver supports the GridFS interface mentioned in Section 2, and the images and enormous videos are exchanged directly between files and the database as streaming. That is, the sensor data entered into the data file of the fog node is directly written into the database, without handling by the variable of the program.

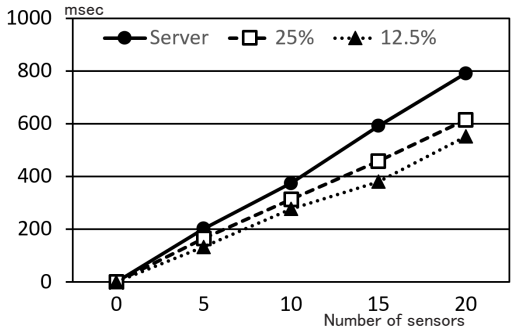

Fig. 4. Elapsed time of sensor data storage.

A. Performance evaluations of data entry process

Firstly, I performed the comparative evaluation of the data entry performance by the following two methods. The first method writes all the entry data into the server directly, so it corresponds to the cloud computing. The second method is based on the proposed method as shown in Fig. 3, and all the entry data is written into the Original database; then, the extracted data is written into the Extraction database. Incidentally, as for the conventional Fog computing, the former is not necessary.

Fig. 4 shows the comparative evaluation results between the first and second method. The horizontal axis of Fig. 4 shows the number of sensors, and in this case, the entry data amount of each sensor is set to 2 MB/sec, which corresponds to one second data amount of videos. Also, the vertical axis shows the elapsed time to save all the sensor data into database. Here, "Server" shows the first method results, and the data is not written into the Original database but is written into Extract database. On the contrary, "25%" shows the first case of the second method, namely the proposed method. In this case, the extraction ratio of data is 25%, that is, 512 KB/sec data are written into the Extraction database; 2 MB/sec data are written into the Original database. Similarly, "12.5%" shows the case of the extraction of data is 12.5%, that is, 256 KB/sec data are written into the Extraction database. Incidentally, in the experiment, the process is repeated five times in one-second cycle, and their average elapsed time is used.

As shown in "Server" of Fig. 4, As for the first method, this elapsed time was 40 msec per sensor, that is, 40 msec per 2 MB. And, as for the second method, that is, the proposed method, in the case where the extraction ratio was 25% or 12.5%, it was more efficient than the first method. In addition, the smaller the extraction ratio was, the more efficient the performance became.

Furthermore, Fig. 5 shows the elapsed time of the proposed method, which was measured separately corresponding to each term of equation (1). Here, the vertical axis and the horizontal axis are the same as in Fig. 4; (1) shows the case where the extraction ratio was 25%; and, (2) shows the case of 12.5%. And, the white part is the elapsed time to write into the Extraction database and corresponds to the second term

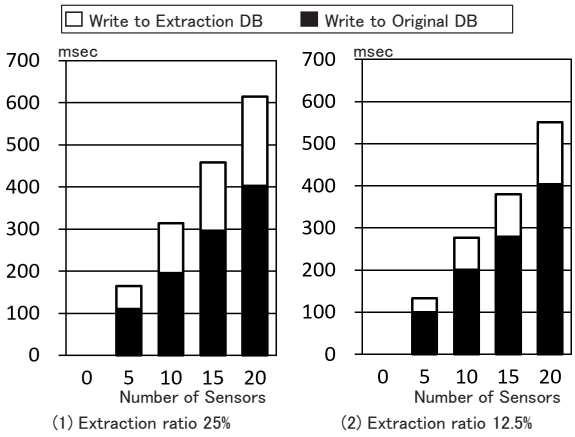

Fig. 5. Breakdown elatsed time of entry.

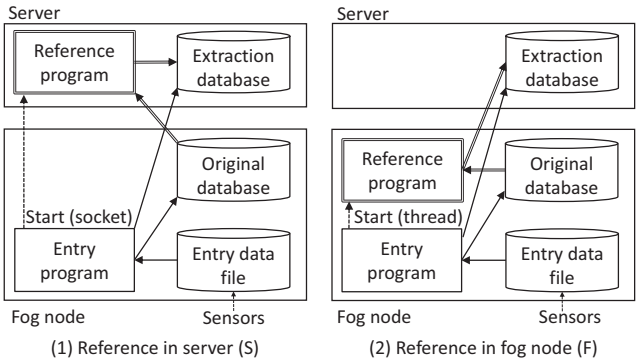

Fig. 6. Two implementation of reference method.

of Equation (1); the black part is to write into the Original database and corresponds to the first term similarly.

The latter was the same elapsed time in both cases irrespective of the extraction ratio, that is, there was the base load to write into the Original database. However, as shown in Equation (3), since this process is performed concurrently in all the fog nodes, this baseload time does not increase even if the number of fog nodes increases. Incidentally, in the case of 20 sensors shown in (1) of Fig. 5, the write time into the Extraction database was 200 msec; the write time into the Original database is 400 msec. So, as for this configuration, since the write data amount into the Original database is eight times larger than as of the Extraction database, the write performance into the Original database was twice of the case to write into the Extraction database.

B. Performance evaluations of reference process

In order to perform the comparative evaluations between two reference ways shown in Section III-B, I added the reference program of each way to the experimental system shown in Fig. 3, which is shown in (1) and (2) of Fig. 6. (1) shows the implementation of the server-side reference shown in Section III-B, in which the Original database in the fog

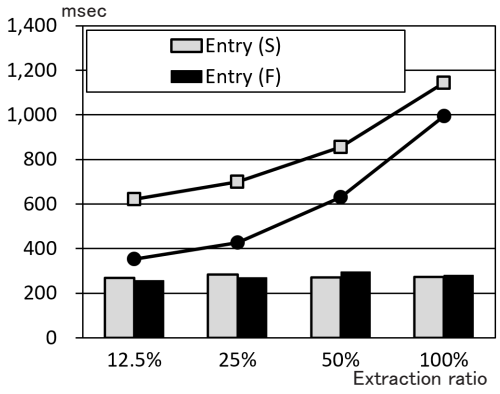

Fig. 7. Elapsed time of data entry and reference.

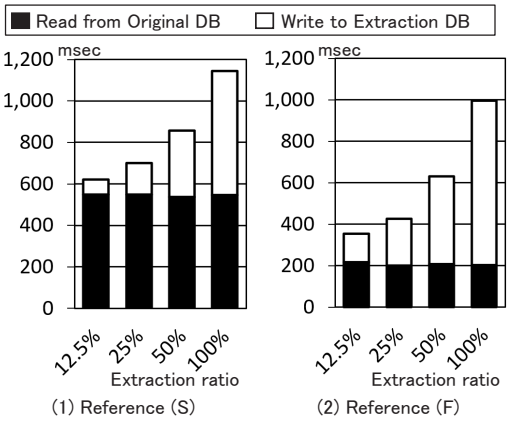

Fig. 8. Breakdown elatsed time of reference.

node is referred from the cloud server and its elapsed time S is expressed by Equation (5). And, in the following graph, I show it by appending "(S)". Similarly, (2) shows the node-side reference, in which the Original database is referred to extract the data to write into the Extraction database at the fog node. Its elapsed time F is expressed by Equation (6), and it is shown by appending "(F)" same as S.

In this experiment, these reference ways were performed concurrently with the data entry shown in Section IV-A, and I adopted the case where the number of sensors was 10 and the extraction ratio was 12.5%. And, as for the reference ways, 10 data of 2 MB, that is, 20 MB data was read from the Original database, and the extraction data of designated ratio was written into the Extraction database.

To perform the data entry and reference concurrently, the latter program was started by the former program after a certain delay. In the experiment of (1) in Fig. 6, in order to perform the reference program on the server side, it was started up by using the Java Socket class from the data entry program on the fog node. Similarly, in the experiment (2), in order to perform both the data entry and reference programs on fog nodes, I used the Thread class to start up the reference program from the entry program.

Fig. 7 shows this experimental results: the bar graph represents the elapsed time of the data entry feature; the line graph represents the time of the data reference feature. As above-mentioned, the gray graph indicated by "S" shows the case where the reference is performed at the server; the black graph indicated by "F" shows the case at the fog node. In addition, the horizontal axis represents the extraction ratio of the data saved into the Extraction database, and the vertical axis represents the elapsed time.

As shown in Fig. 7, in this experiment, the node-side reference (F) was more efficient than the server-side reference irrespective of the extraction ratio. However, as the extraction ratio increased, the elapsed time of both approached. In addition, the data entry time was almost the same as the result shown by the bar graph of Fig. 7, that is, the reference process did not influence the data entry process so much.

Next, Fig. 8 shows the elapsed time of the reference, which was measured separately corresponding with each term of Equation (5) and (6). Here, the vertical axis and the horizontal axis are the same as in Fig. 7; And, (1) shows the results of the server-side reference; (2) shows the one as of the node-side reference. The black part of the graph shows the elapsed time to read from the Original database, which corresponds to the first term of the above-mentioned equations; the white part shows the time to write into the Extraction database, which corresponds to the second term similarly.

As shown in Fig. 8, the elapsed time to read from the Original database is constant irrespective of the extraction rate. And, the node-side reference is about 2.6 times faster than the server-side reference. On the other hand, as for the elapsed time to write into the Extraction database, though the latter is performed at the server, its time is only about 1.5 times faster than one performed at the fog node.

V. DISCUSSIONS

I proposed the data model for Fog computing, which consists of the three levels: the Original database, the Extraction database, and the Result database. By implementing this data model by using the distributed database, each database can be referred among the fog nodes and cloud servers mutually. That is, by equipping the Original database in the fog nodes, which saves the original sensor data, arbitrary original data can be referred additionally even from the cloud server; the extracted data from it can be written into the Extraction database, depending on the needs of the analysis process at the cloud server. So, as mentioned in Section III-A by the case of the surveillance camera, the data to be written into the Extraction database in the cloud server in real time can be limited to only the essential data in ordinary processing; and, the data for the unusual processing such as the incident can be referred from the Original database as necessary. As a result, it is expected that the data amount to be transferred usually to the cloud server can be reduced without being insufficient the necessary data for the analysis.

Next, I discuss the experimental results. First, as for the data entry performance of the proposal method shown in Fig. 4, the smaller the extraction ratio to write into the Extraction database was, the better the performance became. And, the results were obtained in the experiment as mentioned in Section III-B: in the case where the extraction ratio is smaller than 50%, this method has better performance than the cloud computing. For example, Fig. 4 shows in the case where the number of the sensors is 20, the elapsed time of the cloud computing was 800 msec; the one as of the proposed method with 25% extraction rate is 600 msec. Moreover, Fig. 5 shows since Writing to the server is done via the network, it takes more time than writing to the Fog node.

Incidentally, these performances depend on the database performance of the fog node and the cloud server, and especially on the network bandwidth. In this experiment, it is considered that its environment was ideal: I used the terminals having almost the same performance as the cloud server and fog node; the network secured the bandwidth by the switching Hub. And, the network bandwidth is usually more narrow. So, since the data to be transferred via the network is small in the proposed method, it is considered that it becomes more efficient to the cloud computing in the case where the network bandwidth is more narrow. For example, Fig. 5 shows the case of the single fog node. And, in the case of having the plural fog nodes, writing to the Original database that is shown black part of the bar graph in this figure is performed concurrently. Therefore, I consider the more the number of the fog node becomes, the more the performance of proposed method improved compared to the cloud computing; the performance can approach the conventional Fog computing.

Here, since we assume each fog node equips one Original database respectively, its number is equal to the number of the Original databases. On the other hand, since the Extraction database is placed in the cloud server, it is possible to place multiple databases on different virtual servers respectively. In this case, the data transfer to an Extraction database is performed from only the corresponding fog node as shown in Fig. 2. Therefore, by deploying a large number of Extraction databases and fog nodes using a distributed database, it is possible to improve performance by concurrently executing the data transfer.

Second, as for the the Original database reference, I showed the two ways, which were the reference directly from the cloud server and the reference in the fog node, namely the server-side reference and the node-side reference. The difference between them is, as shown in Equation (5) and (6), which of the following is performed via the network: reading from the the Original database, or writing into the Extraction database. And, in this experimental environment, the latter had better performance.

Also in this case, as shown in Equation (5) and (6), similar to the data entry, these performances depend on the database performance of the fog node and cloud server, and on the network bandwidth. However, as for the reference at the fog node, in the case where the plural fog nodes are deployed, readings from the Original databases are performed concurrently at all the fog nodes. Therefore, in the case of the collective reference of the Original databases of many fog nodes, it is considered that the distributed database on the fog nodes is effective.

Moreover, similar to the data entry, by deploying a large number of Extraction databases and fog nodes using a distributed database, this reference is also executed concurrently. And, its performance can also be improved.

Lastly, in the case where many fog nodes are deployed, it is expected that the application of a distributed database to the fog nodes makes data management easier due to the merit of the database itself. The database management system can manage the metadata of the entry data and maintain its security; the availability of the reference process can be improved by using the replication of the distributed database. Incidentally, as shown in Fig. 1, MongoDB is the document-oriented database and does not have the scheme of the database. That is, it can save the entry data with various data structures corresponding to each individual sensor data. In addition, these individual sensor data structure can be managed by using the metadata in the database, and even the binary data such as pictures and videos can be managed by the metadata as shown in Section II.

VI. CONCLUSION

With the progress of IoT, the entry data from various sensors is accumulated in the cloud server and used for various analyses as big data. On the other hand, in order to transfer a large amount of data to the cloud server, there were the problems such as the restriction of network bandwidth, and the delay of feedback control of the sensors. For these problems, Fog computing has been proposed in which the primary processing of the sensor data is performed at the fog node installed near the sensors, and only its processing results are transferred to the cloud server. However, in this method, in the case where the original data of the sensor is required for various analyses in the cloud server, such a data must be transferred additionally.

In this paper, I proposed the distributed data model consisting of three levels for Fog Computing: sensor's original data, extracted data, and its analysis result data. In addition, I showed that this data model can be constructed on the fog node and cloud server by using the distributed database. In this data model, in the case where the original data is required additionally for analysis at the cloud server, arbitrary original data in the fog node can be referred. As a result, it is expected that the data amount to be transferred usually to the cloud server can be reduced. Moreover, I showed the implementation case by using MongoDB, in which the two ways to refer the original data are shown. That is, the first is the way to refer directly from the cloud server; the second is the one to refer to it in the fog node. And, as for the latter, I showed that it is possible to improve efficiency by concurrent processing when deploying many fog nodes.

Currently, the deployment of many various sensors is spreading rapidly. Along with this, the deployment of the fog node itself is expected to spread rapidly. So, future studies will focus on the management functions of the distributed database on a large number of the fog nodes.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number 15K00161.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[3] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Intelligent Systems and Control (ISCO), 2016 10th International Conference on*. IEEE, 2016, pp. 1–8.

[4] M. P. Stevic, B. Milosavljevic, and B. R. Perisic, "Enhancing the management of unstructured data in e-learning systems using mongodb," *Program*, vol. 49, no. 1, pp. 91–114, 2015.

[5] E. Redmond and J. R. Wilson, *Seven databases in seven weeks: a guide to modern databases and the NoSQL movement*. Pragmatic Bookshelf, 2012.

[6] D. R. Rebecca and I. E. Shanthi, "A nosql solution to efficient storage and retrieval of medical images," *International Journal of Scientific & Engineering Research*, vol. 7, no. 2, 2016.

[7] T. Kudo, Y. Ito, and Y. Serizawa, "An application of mongodb to enterprise system manipulating enormous data," *Int, Journal of Informatics Society*, vol. 9, no. 3, pp. 97–108, i2017.

[8] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.

[9] I. Lee and K. Lee, "The internet of things (iot): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431–440, 2015.

[10] S. Hiremath, G. Yang, and K. Mankodiya, "Wearable internet of things: Concept, architectural components and promises for person-centered healthcare," in *Wireless Mobile Communication and Healthcare (Mobihealth), 2014 EAI 4th International Conference on*. IEEE, 2014, pp. 304–307.

[11] K. Hong-yan, "Design and realization of internet of things based on embedded system used in intelligent campus," *IJACT: International Journal of Advancements in Computing Technology*, vol. 3, no. 11, pp. 291–298, 2011.

[12] J. Green, "The internet of things reference model," in *Internet of Things World Forum (IoTWF) White Paper*, 2014.

[13] K. Skala, D. Davidovic, E. Afgan, I. Sovic, and Z. Sojat, "Scalable distributed computing hierarchy: Cloud, fog and dew computing," *Open Journal of Cloud Computing (OJCC)*, vol. 2, no. 1, pp. 16–24, 2015.

[14] K. Banker, *MongoDB in action*. Manning Publications Co., 2011.

[15] M. Inc. The mongodb 3.4 manual. [Online]. Available: https://docs.mongodb.com/manual/ (accessed Dec. 3, 2017)

[16] C. Győrödi, R. Győrödi, G. Pecherle, and A. Olah, "A comparative study: Mongodb vs. mysql," in *Engineering of Modern Electric Systems (EMES), 2015 13th International Conference on*. IEEE, 2015, pp. 1–6.