Implementation of Fog Nodes in the Tree-Based Fog Computing (TBFC) Model of the IoT

Ryusei Chida[1](✉), Yinzhe Guo[1], Ryuji Oma[1], Shigenari Nakamura[1],
Dilawaer Duolikun[1], Tomoya Enokido[2], and Makoto Takizawa[1]

[1] Hosei University, Tokyo, Japan
{ryusei.chida.7n,yinzhe.guo.8m,ryuji.oma.6r}@stu.hosei.ac.jp,
nakamura.shigenari@gmail.com, dilewerdolkun@gmail.com,
makoto.takizawa@computer.org
[2] Rissho University, Tokyo, Japan
eno@ris.ac.jp

Abstract. The IoT (Internet of Things) is so scalable that not only computers like servers but also sensors and actuators installed in various things are interconnected in networks. In the cloud computing model, application processes to process sensor data are performed on servers, this means networks are congested and servers are overloaded to handle a huge volume of sensor data. The fog computing model is proposed to efficiently realize the IoT. Here, subprocesses of an application process are performed on not only servers but also fog nodes. Servers finally receive data processed by fog nodes. Thus, traffic to process sensor data in severs and to transmit sensor data in networks can be reduced in the fog computing model. In this paper, we take the tree-based fog computing (TBFC) model where fog nodes are hierarchically structured in a height-balanced tree. We implement types of subprocesses of fog nodes in Raspbery PI. In experiment of the implemented TBFC model, we show the total execution time of nodes in the TBFC model is shorter than the cloud computing model.

Keywords: IoT (Internet of Things) · Fog computing model · Tree-based fog computing (TBFC) model · Raspberry Pi

1 Introduction

The IoT (Internet of Things) is composed of not only computers like servers and clients but also devices like sensors and actuators which are interconnected in networks [10]. In the cloud computing model [6], every sensor data is transmitted from sensors to servers of clouds in networks. Sensor data is processed by application processes on servers and then servers send actions to actuators. The IoT is more scalable than traditional information systems since a huge number

L. Barolli et al. (Eds.): EIDWT 2019, LNDECT 29, pp. 92–102, 2019.
https://doi.org/10.1007/978-3-030-12839-5_8

of sensors are interconnected and huge amount of sensor data are transmitted in networks. The network is congested due to heavy network traffic of sensor data and servers are also overloaded to process sensor data.

In order to efficiently realize the IoT, the fog computing model [13] is proposed. Here, subprocesses of an application process to process sensor data are performed on not only servers in clouds but also fog nodes while performed only on servers in the cloud computing model [6]. Data obtained by sensors are first transmitted to edge fog nodes. On receipt of sensor data, a fog node processes the sensor data and outputs processed data to another fog node. For example, a fog node obtains an average value of a collection of humidity data collected by sensors and sends only the average value of the humidity data to another fog node. Thus, fog nodes receive and process input data from other fog nodes and send output data obtained by processing the input data to other fog nodes. Servers finally receive data processed by fog nodes and can be relived from the computations done by fog nodes. In addition to the routing function of a router, a subprocess of an application process to process sensor data is installed in a fog node.

The TBFC (Tree-Based Fog Computing) model is proposed to reduce electric energy consumed by fog nodes and servers in the IoT [11,14]. Here, fog nodes are hierarchically structured in a height-balanced tree. A root fog node indicates a cluster of servers. Each non-root fog node has one parent fog node. Each non-leaf node has one or more than one child fog node. An edge fog node is a leaf node of the tree and communicates with sensors and actuators. An application process to be performed on servers to handle sensor data in the cloud computing model is assumed to be a sequence of subprocesses in this paper. Each subprocess receives input data from a preceding subprocess and sends output data to a succeeding subprocess. In the TBFC model, a same subprocess is installed in fog nodes at each level. Thus, each fog node at the same level performs the same subprocess on input data sent by child fog nodes and sends processed output data to a parent fog node. Sensor data from a huge number of sensors are processed by multiple fog nodes in a distributed and parallel manner. The fault-tolerant tree-based fog computing (FTTBFC) model is also proposed to make the TBFC model tolerant of faults of fog nodes [11,12].

In this paper, we implement each fog node of the TBFC model by using a Raspberry Pi [3] computer in this paper. Each subprocess of each fog node is characterized by the computation complexity $O(x)$ or $O(x^2)$ for size x of input data. We show the experiments of the implemented fog nodes of the TBFC model and show that the total execution time fog nodes and a server of the TBFC model is shorter than the cloud computing model.

In Sect. 2, we present the tree-based fog computing (TBFC) model. In Sect. 3, we discuss the implementation of the TBFC model. In Sect. 4, we show experiments of the implemented TBFC model.

2 Tree-Based Fog Computing (TBFC) Model

2.1 Tree of Fog Nodes

The fog computing model of the IoT is composed of devices, fog nodes, and clouds of servers [10]. Each server in a cloud supports applications with computation and storage services like the cloud computing model [6]. There are networks of fog nodes to interconnect devices and clouds. Devices like sensors and actuators are installed in various types of things. In the tree-based fog computing (TBFC) model [14], fog nodes are hierarchically structured in a height-balanced tree. A root node shows a cloud of servers. Each fog node is interconnected with a parent fog node and child fog nodes in networks. Fog nodes at the bottom layer are *edge* fog nodes. Edge fog nodes communicate with child sensors and actuators. Each device, i.e. sensor and actuator, has a parent edge fog node. A sensor collects sensor data and sends the sensor data to an edge fog node. Each edge fog node first collects sensor data from sensors. Each edge fog node processes sensor data and sends processed output data to a parent fog node. A fog node receives input data from child fog nodes and processes the data. Then, the fog node sends the processed output data to the parent fog node. Thus, servers in clouds finally receive processed data from fog nodes. Servers just process data processed by fog nodes and decide on actions to be done by actions. Servers send actions to fog nodes. Fog nodes forward the actions to their child fog nodes and each edge fog node finally sends actions to child actuators.

Figure 1 shows a tree of fog nodes in the TBFC model. Here, f_0 is a root node which denotes a cloud of servers. The root node f_0 has l_0 (≥ 0) child fog nodes f_{00}, f_{01}, ..., f_{0,l_0-1}. Each child fog node f_{0i} has l_{0i} (≥ 0) child fog nodes f_{0i0}, f_{0i1}, ..., $f_{0i,l_{0i}-1}$. Thus, a fog node f_R is f_0 if f_R is a root node, i.e. $R = 0$. If f_R is an i th child fog node of a fog node $f_{R'}$, $f_R = f_{R'i}$, i.e. $R = R'i$ ($i < l_R$). Thus, the label R of a fog node f_R shows a path from a root fog node f_0 to the fog node f_R. $|R|$ shows the length of label R of a fog node f_R. Here, a fog node f_R is at level $|R| - 1$. For example, a root fog node f_0 is at level 0 and a fog node f_{010} is at level 2. On a fog node f_R, a subprocess $p(f_R)$ of an application process is performed. At each layer, a same subprocess is performed on every fog node.

A subprocess $p(f_R)$ of a fog node f_R receives input data d_{R0}, d_{R1}, ..., d_{R,l_R-1} from child fog nodes f_{R0}, f_{R1} ..., f_{R,l_R-1}, respectively. Let D_R be a set of input data d_{R0}, ..., d_{R,l_R-1} of the fog node f_R. Then the input data is processed and output data d_R is generated. The output data d_R is sent to a parent fog node pt_{f_R}.

2.2 Subprocesses on Fog Nodes

An application process p is assumed to be a sequence $\langle p_0, p_1, ..., p_{h-1} \rangle$ of subprocesses. In the TBFC model, fog nodes are structured in a height-balanced tree of fog nodes with height h. All the subprocesses p_0, p_1, ..., p_{h-1} are performed on servers in the cloud computing model. Only the subprocess p_0 is performed on a root node, i.e. server cloud f_0. The subprocess p_0 is a *root* subprocess. The other subprocesses p_1, ..., p_{h-1} are performed on different fog nodes. The subprocess p_{h-1} is first performed on edge fog nodes of level $h-1$ by receiving data from sensors. The subprocess p_{h-1} is an *edge* subprocess. The output data of the edge subprocess p_{h-1} is sent to the succeeding subprocess p_{h-2}, which is performed on each of fog nodes of level $h-2$. Thus, a subprocess p_i is performed on fog nodes of level i and sends output data to a succeeding subprocess p_{i-1} on fog nodes of level $i-1$. The lower layer, the more amount of input data are sent from the underlying layer but the more number of fog nodes since the TBFC model is tree-structured and the output data of each fog node is generally smaller than the input data. Hence, the processing load of each fog node is equalized.

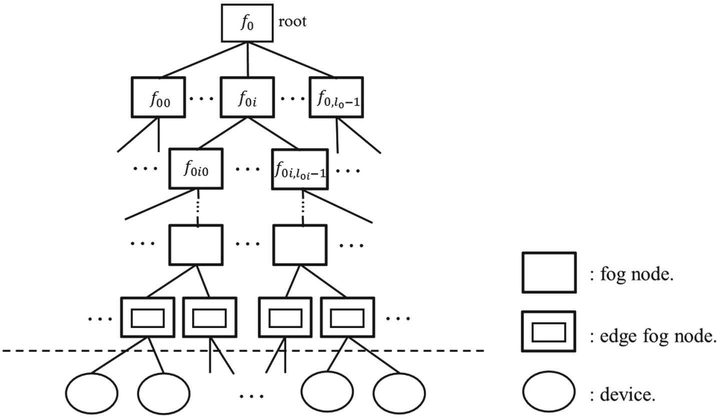

Fig. 1. TBFC model.

In the cloud computing model [6], a fog node is just a router which does only the routing function. In each fog node, not only the routing function but also a subprocess of an application process are performed in the fog computing model. Sensors send every sensor data to servers in a cloud.

There are types of subprocesses characterized in terms of computation complexity. In this paper, we consider the following types of subprocesses to be performed on a fog node f_R to handle input data D_R of size x $(= |D_R|)$ [12]:

1. $O(x)$, e.g. subprocess to calculate an average value of input data D_R.
2. $O(x^2)$, e.g. subprocess to join multiple input data d_{R0}, ..., d_{R,l_R-1} in input data D_R.

For example, a fog node f_R just selects some data in input data D_R of size x $(= |D_R|)$, e.g. a maximum value is selected in a collection D_R of input data from child fog nodes. Here the computation complexity of the subprocess is $O(x)$. The computation complexity of a fog node f_R to merge sorted input data is $O(x)$. The computation complexity of a fog node f_R which joins multiple input data is $O(x^2)$ where multiple input data D_R which have the same attribute values are concatenated.

3 Implementation of the TBFC Model

We discuss the implementation of fog nodes in the TBFC model. Each fog node f_R is implemented in a Raspberry Pi 3 Model B [3] computer with a CPU ARM Cortex-A53, one [GB] memory, and 32 [GB] SD storage. The Raspbian 8.0 [3] is used as an operating system of a fog node. A subprocess of an application process to be performed on each fog node is implemented in C language. Fog nodes are interconnected in a Gbit local area network (LAN). Fog nodes communicate with one another by using the protocol UDP [5]. A *record* is a unit of communication among fog nodes where data is stored. Each record is composed of attributes. Each fog node f_R has one UDP socket, US. A fog node f_R receives records from child fog nodes f_{R0}, ..., f_{R,l_R-1} and sends records at the UDP socket US.

We consider the following types of subprocesses to be performed on each fog node f_R:

1. In a fog node f_R, an aggregate value, e.g. average value of input data D_R is obtained and the aggregate data d_R is output by an *aggregate* subprocess [Fig. 2(1)]. The computation complexity of the fog node f_R is $O(x)$ for size x $(= |D_R|)$ of input data D_R.
2. In a fog node f_R, sorted input data d_{R0}, d_{R1}, ..., f_{R,l_R-1} are merged into a sorted data d_R by a *sort* subprocess [Fig. 2(2)]. The computation complexity of the fog node f_R is $O(x)$.
3. In a fog node f_R, multiple input data d_{R0}, d_{R1}, ..., $d_{R,l_R-1}$are joined to one output data d_R by a *join* subprocess [Fig. 2(3)]. The computation complexity of the fog node is $O(x^2)$.

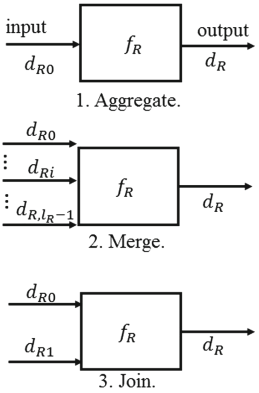

Fig. 2. Types of subprocesses on fog nodes.eps

A fog node f_R receives a record of input data d_{Ri} from each child fog node f_{Ri} and stores the record d_{Ri} in a receipt queue RQ_i. Each record d_{Ri} is composed of attribute values, i.e. tuple $\langle v_1, ..., v_l \rangle$ $(l > 1)$. On receipt of each record d_{Ri}, a cell c is dynamically allocated in a fog node f_R. The record d_{Ri} is stored in the cell c and the cell c is enqueued into the receipt queue RQ_i. While dequeueing a top record d_{Ri} from each receipt queue RQ_i, records of input data d_{R0}, d_{R1}, ..., d_{R,l_R-1} are processed and a record of output data d_R is generated by performing a subprocess on the input data D_R. The output record d_R is enqueued into an output queue SQ. A top record d_R is dequeued from the output queue SQ and sent to the parent fog node of f_R by using UDP.

Each queue Q is implemented in data structure as shown in Fig. 3. Each queue Q is composed of a control block CBC and doubly linked cells. The variable *no* of the data structure CBC shows the number of cells in the queue Q. The *top* and *tail* fields of the CBC block denote pointers to the top and tail cells of the queue Q. On receipt of a record d_{Ri} from a child fog node f_{Ri}, one cell c is created by a *malloc* system call [7] and the record d_{Ri} is stored in the cell c. Then, the cell c is enqueued in the receipt queue RQ_i, i.e. stored as the tail cell of the receipt queue RQ_i. Cells are linked in bidirectional pointers, *next* and *prior*. The next and prior pointers of a cell c denote cells following and preceding the cell c, respectively.

Each queue Q is manipulated through the following functions:

1. struct CBC *iniqueue( );
2. enqueue (struct CBC **cbc*, struct CELL **c*);
3. struct CELL *dequeue (struct CBC **cbc*);
4. struct CELL *topqueue (struct CBC **cbc*);

First, a control block *cbc* is created by the function *iniqueue*(), i.e. $cbc = iniqueue()$. A cell c is dequeued from the queue *cbc* by the function $c = dequeue\ (cbc)$. A cell c is enqueued to the queue *cbc* by the function *enqueue* (*cbc*, *c*). A top cell c in the queue *cbc* is found by $c = topqueue\ (cbc)$.

In this paper, every subprocess to be performed on each fog node is implemented on the Raspbian operating system in C language. A subprocess on each fog node f_R communicates with each child fog node and a parent fog node by using the UDP protocol [5].

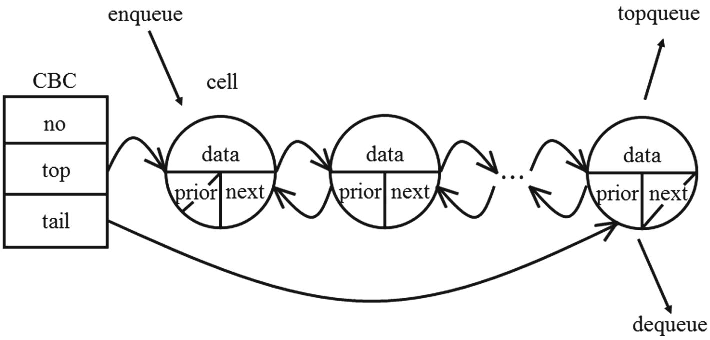

Fig. 3. Queue.

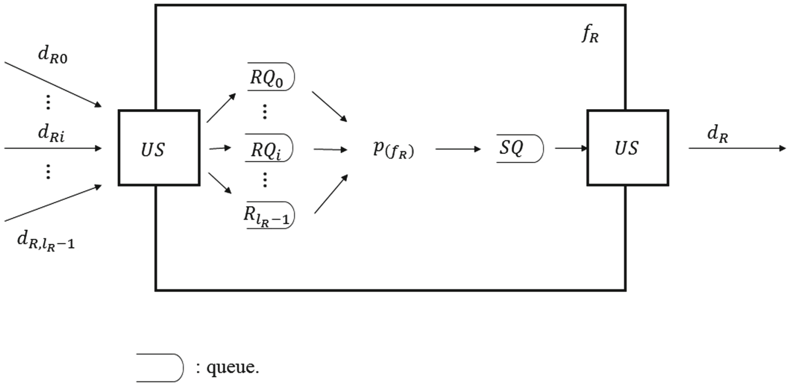

Fig. 4. Fog node.

4 Experiment

4.1 Implemented TBFC Model

We present the experiment of the implemented TBFC model. There are four sensors s_0, s_1, s_2, and s_3, and seven fog nodes f_{00}, f_{000}, f_{001}, f_{0000}, f_{0001}, f_{0010}, and f_{0011} in a tree. The sensors, fog nodes, and sever are interconnected in a Gbit LAN. One root node f_0 is in a cluster. As shown in Fig. 5, the root fog node f_0 is a server in the cloud. The root node f_0 has a single child fog node f_{00}. The fog node f_{00} has a pair of child fog nodes f_{000} and f_{001}. The fog nodes f_{000} and f_{001} of level 2 have pairs of child fog nodes f_{0000} and f_{0001}, and f_{0010} and f_{0011}, respectively. The four fog nodes f_{0000}, f_{0001}, f_{0010}, and f_{0011} are edge fog nodes at level 3, which communicate with sensors s_0, s_1, s_2, and s_3, respectively.

Each of the fog nodes and sensors is implemented in a Raspberry PI BM model computer [3]. A pair of the sensors s_0 and s_1 collect temperature data. A pair of the sensors s_2 and s_3 collect humidity data. Each sensor gets sensor data every one second and sends the data to a parent edge fog node.

The sever f_0 is a PC with a CPU Intel Xeon X3430, 16 [GB] memory, and 2 [TB] HDD, whose operating system is CentOS 7 [1]. In the server f_0, a Sybase [4] database is supported to store data obtained from the fog node f_{00}. In the TBFC model, the *store* subprocess is implemented in C language with transact SQL [4].

In the experiment, each edge fog node is equipped with one sensor as shown in Fig. 4. A pair of the edge fog nodes f_{0000} and f_{0001} collect temperature data from the sensors s_0 and s_1, respectively. Another pair of edge fog nodes f_{0010} and f_{0011} collect humidity data from the sensors s_2 and s_3, respectively. A process of a sensor to collect sensor data is realized in Python [2]. Sensor data is collected by the polling mechanism every one second. A record of collected sensor data is sent by each sensor to an edge fog node, which is composed of *time* and *value*. This means, the *value* is sensor data, i.e. temperature and humidity, which is obtained at the *time*. That is, temperature data is collected by a pair of the sensors s_0 and s_1 and humidity data is collected by another pair of the sensors s_2 and s_3 at *time*. The sensors s_0, s_1, s_2, and s_3 send records of sensor data to the edge fog nodes f_{0000}, f_{0001}, f_{0010}, and f_{001}, respectively, every one second.

Each edge fog node f_{00ij} receives sensor data from a sensor every one second. Then, the edge fog node f_{00ij} calculates an average value of sensor data, temperature or humidity data collected from the sensors for every one minute (i, j = 0, 1). A subprocess *aggregate* is performed by each edge fog node f_{00ij} to obtain an average value from input data. Then, the edge fog node f_{00ij} sends the output data d_{00ij} to the parent fog node f_{00i}.

A parent fog node f_{00i} receives a pair of input data d_{00i0} and d_{00i1} from child fog nodes f_{00i0} and f_{00i1}, respectively. The parent fog nodes f_{000} and f_{001} collect temperature data and humidity data from child fog nodes, respectively. A subprocess *merge* is performed on each fog node f_{00i}. Then, a pair of the input data d_{00i0} and d_{00i1} are merged into the output data d_{00i}. If values of input data d_{00i0} and d_{00i1}, whose *time* is the same, are received, the output data d_{00i} is the average value of the input data d_{00i0} and d_{00i1}. Here, the output data d_{000} of temperature is sorted in *time*. The output data d_{001} of humidity is also sorted in *time*. The fog node f_{00i} sends the output data d_{00i} to the fog node f_{00} (i = 0, 1).

The fog node f_{00} receives input data from the child fog nodes f_{000} and f_{001}. A pair of input data $d_{000} = \langle v_{000}, t_{000}\rangle$ and $d_{001} = \langle v_{001}, t_{001}\rangle$ are joined, i.e. concatenated by the fog node f_{00} into one output data d_{00}. In the output data d_{00}, temperature data v_{000} and humidity data v_{001} whose time is the same, i.e. $t_{000} = t_{001} = t$ are concatenated to a record $\langle t, v_{000}, v_{001}\rangle$. Thus, a subprocess *join* is performed on the fog node f_{00}. The fog node f_{00} sends the output data d_{00} to the root node f_0.

The root fog node f_0 is a server which receives input data $d_{00} = \langle t, v_{001}, v_{001}\rangle$ from the fog node f_{00}. The server f_0 stores the data d_{00} to a table Data (time, temperature, humidity) in the database DB_0 by SQL insert [7] once the server f_0 receives the data. The database DB_0 is implemented in Sybase [4]. A subprocess *store* is performed on the root node f_0.

In the cloud computing model, the sensors s_0, s_1, s_2, and s_3 are directly interconnect with a sever f_0 as shown in Fig. 5. Sensor data from the sensors s_0, s_1, s_2, and s_3 are sent to the server f_0 by using UDP in a Gbps LAN. All the *aggregate*, *merge*, *join*, and *store* subprocesses are performed on the server f_0. Every sensor data sent by the sensors is processed by a sequence of the *aggregate*, *merge*, *join*, and *store* subprocesses on the server f_0.

4.2 Experiment

We measure total execution time TET [sec] of all the fog nodes and the server since the sensors s_0, s_1, s_2, and s_3 send temperature and humidity data to the edge fog nodes until the server f_0 stores the data to the database DB_0 in the cloud computing model and in the TBFC model. In order to measure the total execution time TET, one fog node f_c is used as shown in Figs. 5 and 6. The fog

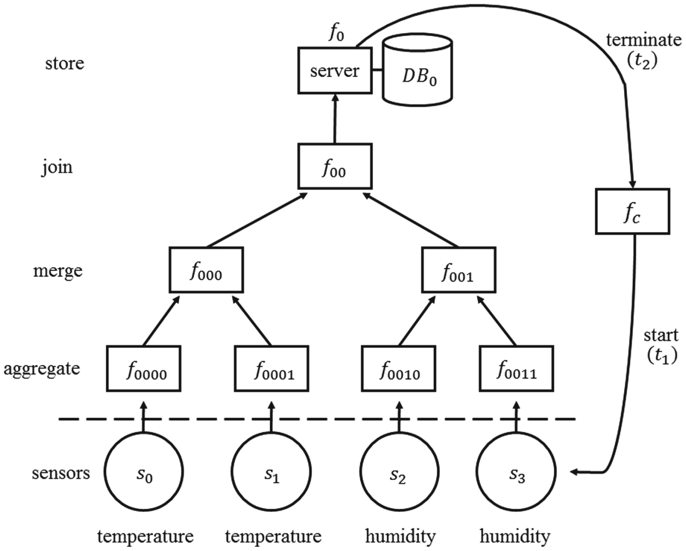

Fig. 5. Experiment of TBFC model.

node f_c sends a *start* message to every sensor at time t_1 and then the sensors start sending sensor data. If every sensor data is stored in the database DB_0, the server f_0 sends a *termination* message to the fog node f_c. The fog node f_c receives the *termination* massage at time t_2. Here, the total execution time TET is $t_2 - t_1$ [sec].

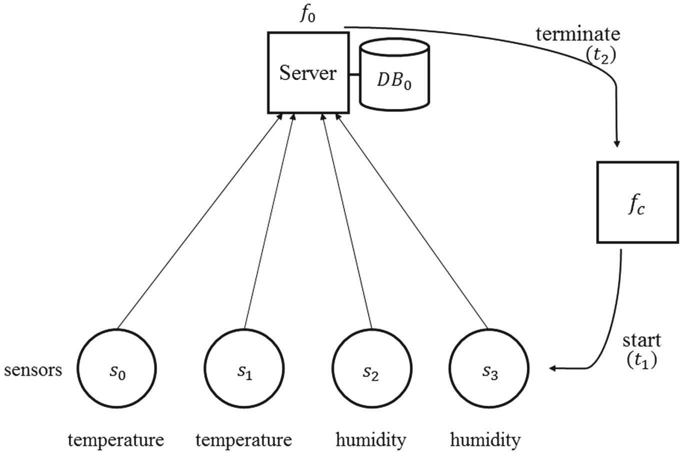

Fig. 6. Cloud computing model.

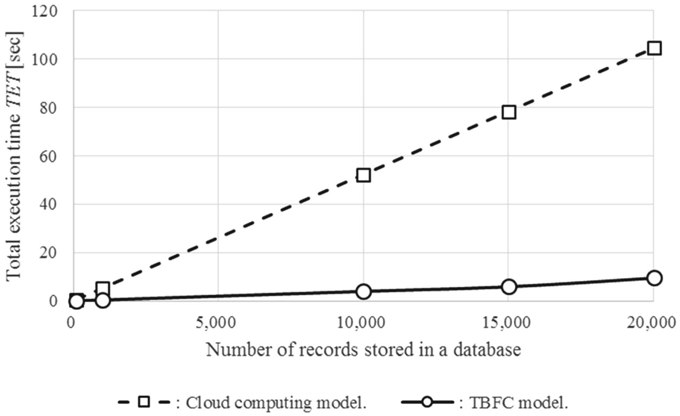

Fig. 7. Total execution time.

Figure 7 shows the total execution time TET of the TBFC model and the cloud computing model for number rn of records stored in the database DB_0. The total execution time TET of the TBFC model and the cloud computing model linearly increases as the number rn of records increases. The total execution time TET of the TBFC model is shorter than the cloud computing model. For example, the total execution time TET of the TBFC model is only 7% and 9% of the cloud computing model, respectively, for $rn = 10,000$ and $rn = 20,000$. This experiment shows the IoT can be efficiently realized by the TBFC model.

5 Concluding Remarks

The fog computing model is useful to efficiently realize the IoT since processes and data are distributed to not only servers but also fog nodes. Then, traffic of servers and networks can be reduced in the fog computing model. In this paper, we discussed the implementation of each fog node in the tree-based fog computing (TBFC) model by a Raspberry Pi 3 Model B computer. We implemented the subprocesses of computation complexity $O(x)$ and $O(x^2)$ for size x of input data, to be performed by each fog node. We showed the experiment of the implemented TBFC model in Raspberry PI.

Here, four sensors, seven fog nodes, and a server are hierarchically structured in a height-balanced tree. In the evaluation, the total execution time of the TBFC model is about 90% to 95% shorter than the cloud computing model. We showed the IoT can be efficiently realized in the TBFC model.

In the IoT, it is critical to reduce the total electric energy consumption [11,12]. We are now evaluating the TBFC model in terms of electric energy [8,9] consumed by fog nodes and servers.

Acknowledgements. This work was supported by JSPS KAKENHI grant number 15H0295.

References

1. The centos linux distribution (centos linux). https://www.centos.org/
2. Python. https://www.python.org/downloads/release/python-2713/
3. Raspberry Pi 3 model B. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
4. Sybase. https://www.sap.com/products/sybase-ase.html
5. Comer, D.E.: Internetworking with TCP/IP, vol. 1. Prentice Hall, Englewood Cliffs (1991)
6. Creeger, M.: Cloud computing: an overview. Queue **7**(5), 3–4 (2009)
7. Date, C.J.: An Introduction to Database System, 8th edn. Addison Wesley, Reading (2003)
8. Enokido, T., Ailexier, A., Takizawa, M.: An extended simple power consumption model for selecting a server to perform computation type processes in digital ecosystems. IEEE Trans. Industr. Inf. **10**(2), 1627–1636 (2014)
9. Enokido, T., Ailexier, A., Takizawa, M.: A model for reducing power consumption in peer-to-peer systems. IEEE Syst. J. **4**(2), 221–229 (2010)
10. Hanes, D., Salgueiro, G., Grossetete, P., Barton, R., Henry, J.: IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, Indianapolis (2018)
11. Oma, R., Nakamura, S., Duolikun, D., Enokido, T., Takizawa, M.: Evaluation of an energy-efficient tree-based model of fog computing. In: Proceedings of the 21st International Conference on Network-Based Information Systems (NBiS-2018), pp. 99–109 (2018)
12. Oma, R., Nakamura, S., Duolikun, D., Enokido, T., Takizawa, M.: Fault-tolerant fog computing models in the IoT. In: Proceedings of the 13th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2018) (Accepted, 2018)
13. Oma, R., Nakamura, S., Enokido, T., Takizawa, M.: An energy-efficient model of fog and device nodes in IoT. In: Proceedings of IEEE the 32nd International Conference on Advanced Information Networking and Application (AINA-2018), pp. 301–306 (2018)
14. Oma, R., Nakamura, S., Enokido, T., Takizawa, M.: A tree-based model of energy-efficient fog computing systems in IoT. In: Proceedings of the 12th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS-2018), pp. 991–1001 (2018)