International Journal of Grid and Distributed Computing
Vol. 11, No. 7 (2018), pp.23-30
http://dx.doi.org/10.14257/ijgdc.2018.11.7.03

Hybrid Deep Neural Network based Performance Estimation Method for Efficient Offloading on IoT-Cloud Environments

Yunsik Son[1], Seman Oh[1] and Yangsun Lee[2*]

[1]*Dept. of Computer Science and Engineering, Dongguk University, 3-26 Pil-dong, Jung-gu, Seoul 100-715, Korea*
[2]*Dept. of Computer Engineering, Seokyeong University, 16-1 Jungneung-Dong, Sungbuk-Ku, Seoul 136-704, Korea*
sonbug@dongguk.edu, smoh@dongguk.edu yslee@skuniv.ac.kr

Abstract

The IoT-Cloud virtual machine system is a cloud-based execution solution for IoT devices with offloading techniques that delegate tasks requiring high computing power from low-performance IoT devices to a high-performance cloud environment as a service. The IoT devices with the IoT-Cloud virtual machine system can perform complex tasks using the computing power of high-performance cloud. The offloading technique can reduce the execution performance depending on the workload of the IoT devices and the clouds. Therefore, it is necessary to decide offloading execution considering the workload of the IoT devices and the clouds.

In this paper, CPU utilization trend, which is one of the workload indices, is predicted through deep learning in order to decide offloading execution considering the workload of the IoT devices and clouds. In this paper, we present four CPU usage models and introduce a technique for predicting server load based on hybrid deep neural network. The predicted CPU utilization trend is indicative of future CPU utilization information and is therefore an indicator for offloading execution decisions. Through experiments, we confirmed that the proposed method estimates the load of the model very similar, and it can apply the offloading adaptively according to the load of the server.

Keywords: *IoT Devices, Virtual Machine, Offloading, Performance Prediction, Deep Neural Network*

1. Introduction

The IoT-Cloud virtual machine system [1] uses an offloading technique to perform tasks with high computational complexity by providing high-performance cloud server computing power to low-performance IoT devices. The offloading execution of the current IoT-Cloud virtual machine system has a simple structure in which offloading execution is determined when the computational complexity of the task analyzed through profiling exceeds a predetermined value. However, since the offloading technique may reduce the performance depending on the workload and the network performance of the IoT devices and the cloud server, the offloading operation should be decided considering the workload of the IoT devices and the cloud server.

This paper predicted CPU utilization trend, which is one of the workload indices, through deep learning. The predicted CPU utilization trend is indicative of future CPU utilization information and is therefore an indicator for offloading execution decisions.

Received (March 19, 2018), Review Result (June 15, 2018), Accepted (June 20, 2018)
* Corresponding Author

ISSN: 2005-4262 IJGDC

2. Related Works

2.1. LWVM (Light-Weighted Virtual Machine)

The IoT-Cloud VM is a stack-based virtual machine designed to run on IoT devices under low computational performance. Figure 1 shows the system configuration of the IoT-Cloud VM [5].

The Light-Weighted IoT Virtual Machine is a stack-based virtual machine that was designed to execute on low computing powered IoT devices by the cloud-based offloading method. Moreover, it was designed for small computing devices that have restricted resources. The system configuration of the proposed Light-Weighted IoT Virtual Machine is comprised of the execution layer, offloading layer, runtime environment layer, portable library layer, and JIT (Just-In-Time) compiler.

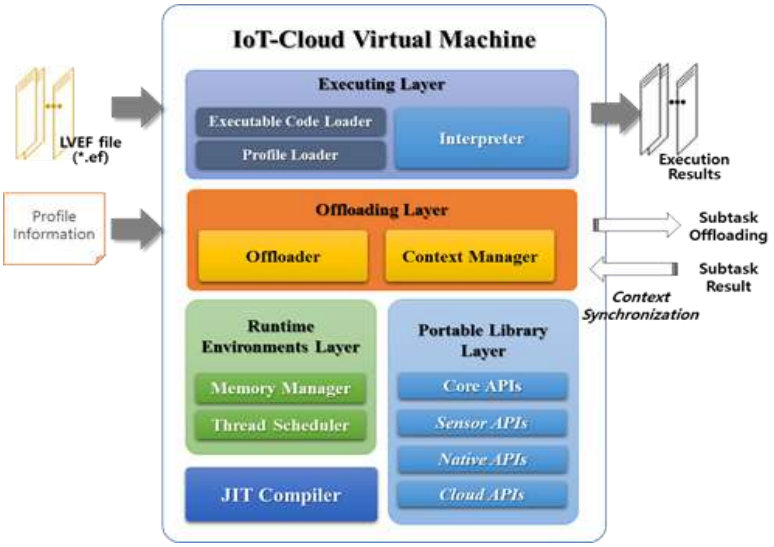


Figure 1. System Configuration of the IoT-Cloud VM (LWVM, Light-weighted Virtual Machine)

2.2. IoT-Cloud Fusion Virtual Machine System

The IoT-Cloud fusion virtual machine system is designed to support downloading and executing application programs without platform dependency on various IoT devices [5].

The system consists of four main parts; compiler, assembler, profiler and IoT-Cloud virtual machine. It is designed in a hierarchal structure to minimize the burden of the retargeting process. Figure 2 shows a system configuration of the IoT-Cloud fusion virtual machine system.

The IoT-Cloud Virtual Machine is a stack-based virtual machine solution with cloud offloading technology, loaded on IoT devices, which allows dynamic application programs to be downloaded and run platform independently with high computing performance [7, 8]. The VM is designed to use an intermediary language, RSIL (Reduced Smart Intermediate Language), which is capable of accommodating both procedural and object-oriented languages. It has the advantage of accommodating languages such as C/C++, Java, and Objective-C used in the iOS, which is currently used by a majority of developers.

The LWVM system consists of a compiler that compiles the application and generates a LVAF (Light-weighted Virtual machine Assembly Format) file composed of RSIL code, an assembler that converts the LVAF file to LVEF (Light-weighted Virtual machine Executable Format), and the IoT-Cloud VM that accepts and executes the LVEF file.

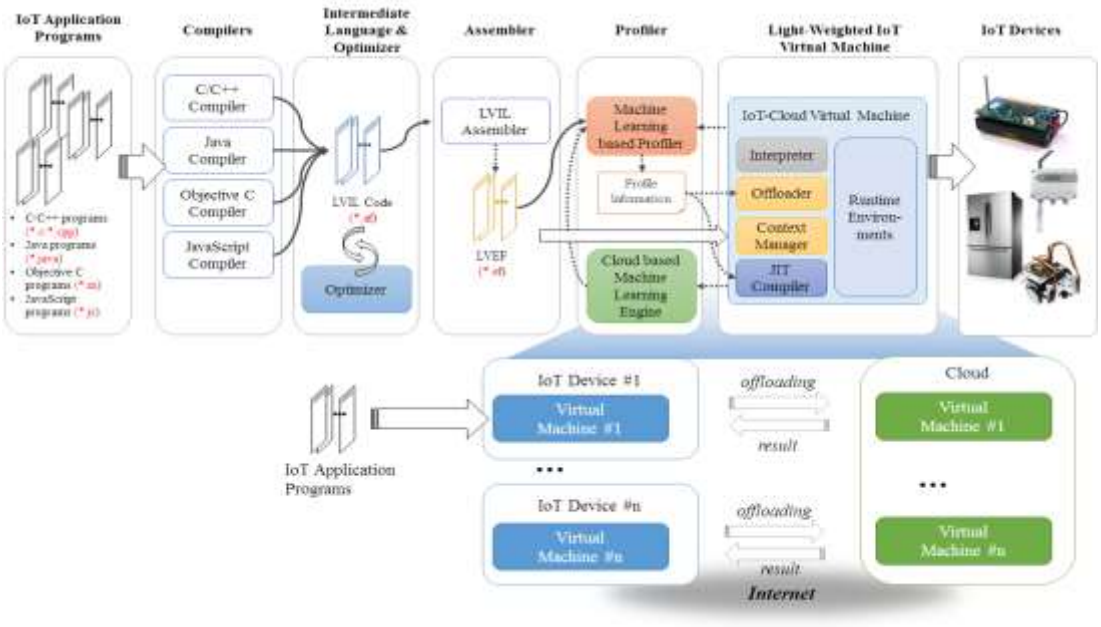


Figure 2. Configuration of the IoT-Cloud Fusion Virtual Machine System

2.3. TreNet

TreNet is a neural network for trend prediction of time series data. It is a hybrid neural network combining LSTM (Long Short-Term Memory), CNN (Convolutional Neural Network) and Feature Fusion layer [2]. In the LSTM layer, time series data consisting of the current trend slope (l_k) and persistence (s_k) is input to the CNN layer, raw data sets are input to learn the dependency of the current trend and pattern transition point, To predict the forecast slope (l_k ') and persistence (s_k') of future trends. Figure 3 shows the structure of TreNet.

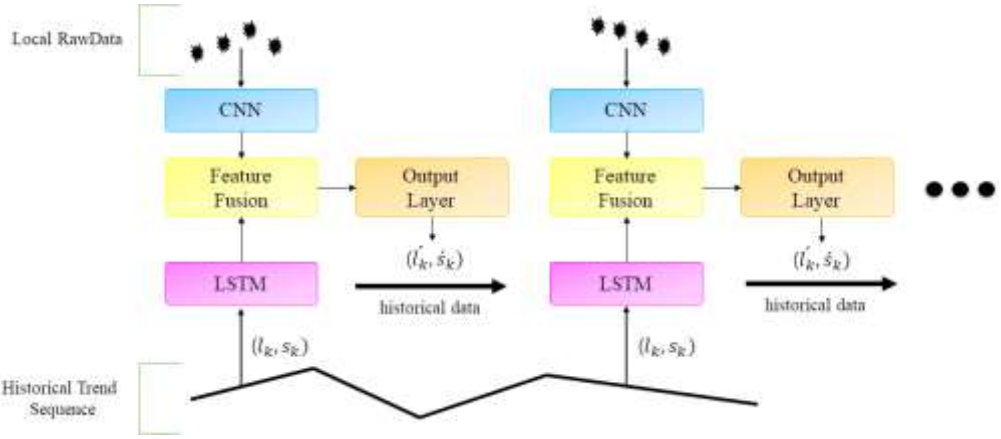


Figure 3. Structure of the TreNet

3. Supervised Learning Model for Performance Load Prediction

The CPU utilization trend of this paper is studied using TreNet, and the overall structure is composed of data set loader, data processor, and TreNet. Figure 4 shows the overall structure of the learning machine for trend prediction. Data processing is the process of processing a set of CPU utilization data into a form suitable for trend learning. The CPU utilization data set loaded into memory in the data set loader is processed by the

data processor into a local data set, a slope data set, and a result data set. The local dataset is used to grasp the pattern conversion point in a form that is normalized with the utilization distribution and bundled with the specified window size.

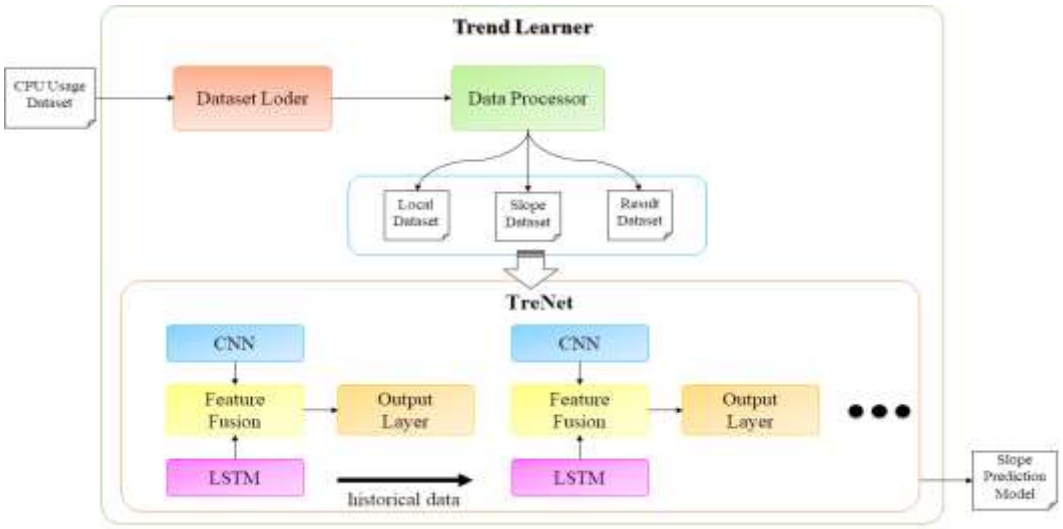


Figure 4. Structure of the Learning Model for Performance Load Prediction

The slope data set is a slope change data set of the CPU utilization distribution and represents historical trend information from past to present. The data processor derives a linear function that minimizes the error in the CPU utilization distribution using the method of Least Squares to extract the gradient change data from the CPU utilization distribution.

The result dataset is a set of data used to learn the results of the map learning and is processed into a set of gradients indicating the future at the input data point in accordance with the learning purpose of this paper.

Since CPU utilization trends are real-time information, it is impossible to predict sustainability even though the slope can be obtained. Therefore, TreNet of this paper learns only the slope except the persistence of the trend. The map learning of this paper proceeds in the following order.

(1) The local dataset is input to the CNN layer of TreNet while the LSTM layer is input to the tilt dataset.

(2) The CNN and LSTM layers learn the dependencies in the input data set and send the results to the Feature Fusion layer.

(3) The Feature Fusion layer fuses the results of the CNN and LSTM layers to learn the dependence of the slope and the pattern transition point.

(4) The Output Layer learns to derive the predictive slope through the dependency relationships and result datasets learned in the Feature Fusion layer.

4. CPU Utilization Modeling and Experimental Results

The CPU utilization pattern is necessary to model the real-world CPU utilization distribution. The CPU utilization pattern of this paper analyzed and classified the CPU utilization data sets collected for the NAB (Numenta Anomaly Benchmark).

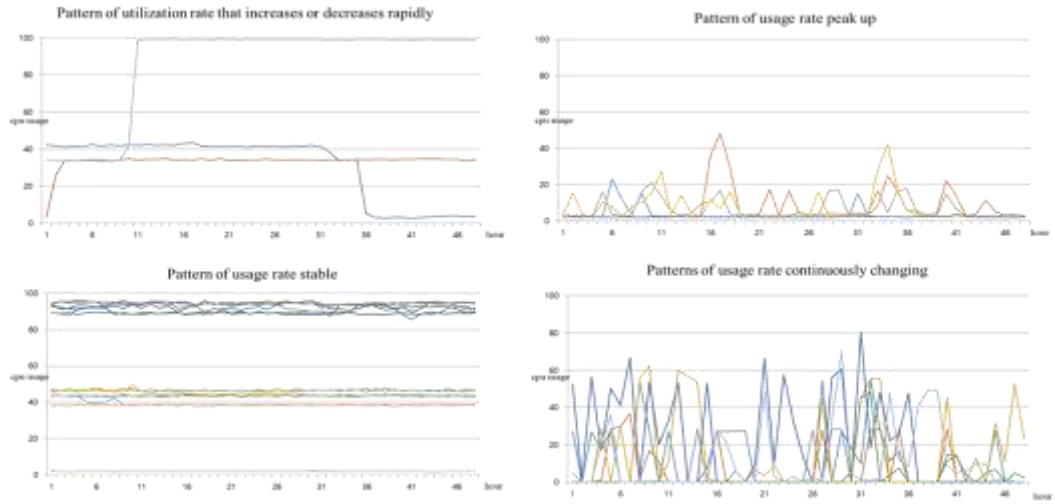


Figure 5. CPU Utilization Classified by Patterns

The distribution of CPU usage in the dataset is stable after the surge in utilization rate, stable after steep decline in utilization rate, stable in utilization rate, and stable in utilization. Peak value is classified into patterns in which the utilization rate continuously changes. Figure 5 shows the patterns classified by analyzing the CPU utilization distribution.

The CPU utilization distribution of this paper is modeled by using the M / M / 1 queuing system which follows the time - varying Poisson process [12, 13].

The M / M / 1 queuing system is not suitable for modeling CPU utilization that changes in real time because the arrival rate of work is fixed. On the other hand, the time-varying M / M / 1 queuing system is suitable for modeling the CPU utilization rate, which is a workload changing in real time because the arrival rate of work varies with time. The arrival rate of work for each pattern defined is defined as follows.

Patterns in which the arrival rate is constantly changing:

$$\begin{array}{ll}
\lambda(t) = 0.05172t - 0.00554 & \text{for } 0.0 \leq t < 1.0 \\
\lambda(t) = 0.20536t - 0.15918 & \text{for } 1.0 \leq t < 1.5 \\
\lambda(t) = -0.0846t + 0.27576 & \text{for } 1.5 \leq t < 2.0 \\
\lambda(t) = -0.14484t + 0.39624 & \text{for } 2.0 \leq t < 2.5 \\
\lambda(t) = 0.2688t - 0.63786 & \text{for } 2.5 \leq t < 3.0 \\
\lambda(t) = -0.31804t + 1.12266 & \text{for } 3.0 \leq t < 3.5 \\
\lambda(t) = 0.098t - 0.33348 & \text{for } 3.5 \leq t < 4.0 \\
\lambda(t) = 0.01476t - 0.00052 & \text{for } 4.0 \leq t < 4.5 \\
\lambda(t) = 0.15328t - 0.62386 & \text{for } 4.5 \leq t < 5.0
\end{array}$$

Work arrival rate is stable and upward pattern:

$$\begin{array}{ll}
\lambda(t) = 0.04 & \text{for } 0 \leq t < 2 \\
\lambda(t) = 0.1t - 0.16 & \text{for } 2 \leq t < 3 \\
\lambda(t) = 0.14 & \text{for } 3 \leq t
\end{array}$$

Work arrival rate is stable and downward pattern:

$$\begin{array}{ll}
\lambda(t) = 0.14 & \text{for } 0 \leq t < 2 \\
\lambda(t) = -0.1t + 0.34 & \text{for } 2 \leq t < 3 \\
\lambda(t) = 0.04 & \text{for } 3 \leq t
\end{array}$$

Work arrival rate is stable and peak pattern:

$$\begin{array}{ll}
\lambda(t) = 0.04 & \text{for } 0 \leq t < 2 \\
\lambda(t) = 0.1t - 0.16 & \text{for } 2 \leq t < 3 \\
\lambda(t) = -0.1t + 0.44 & \text{for } 3 \leq t < 4 \\
\lambda(t) = 0.04 & \text{for } 4 \leq t
\end{array}$$

The results are verified using a simulator to verify that the modeled CPU utilization distribution is normally modeled according to the defined arrival rate of work. Figure 3 shows the simulation results.

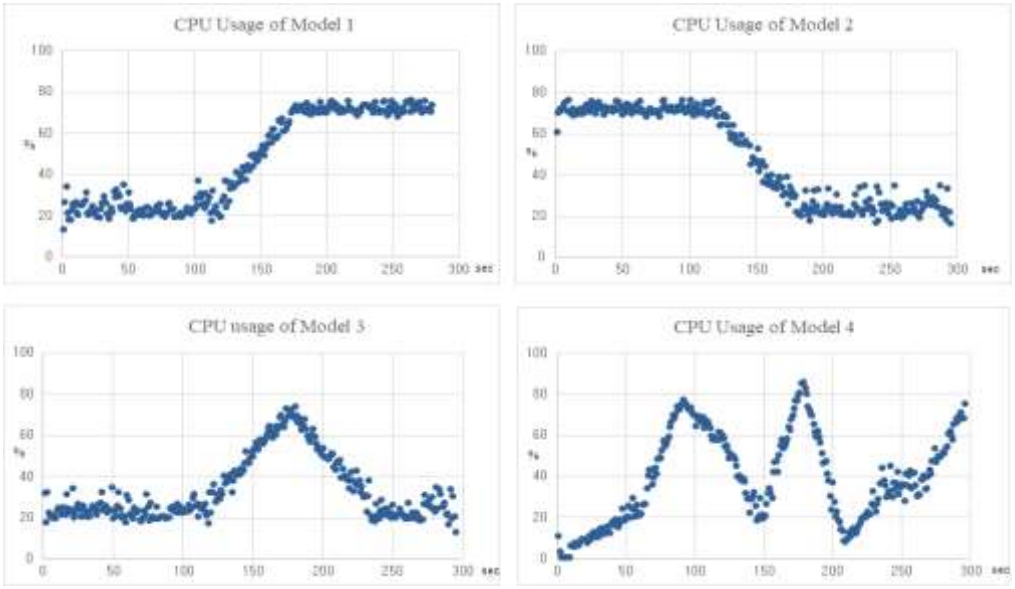
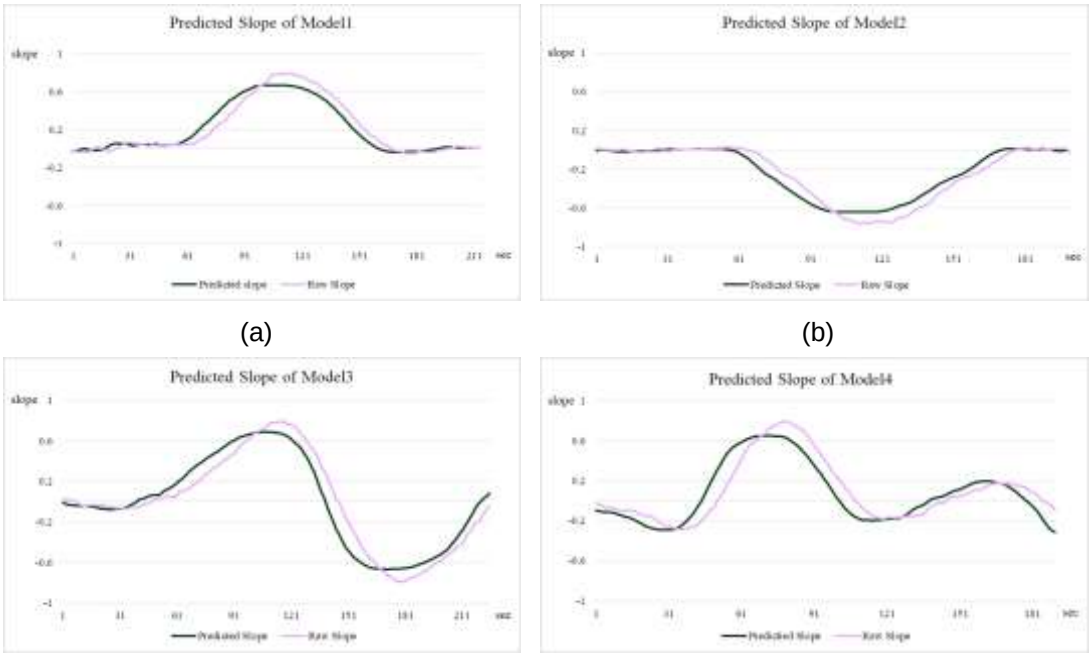


Figure 6. CPU Modeling Simulation Results

The results are verified using a simulator to verify that the modeled CPU utilization distribution is normally modeled according to the defined arrival rate of work. Figure 5 shows the simulation results.



(a) (b)

(c) (d)

Figure 7. Estimation Results for Server's Load using Hybrid Deep Neural Network

Figure 7 is a comparison of the CPU usage distribution modeled using the time-varying M / M / 1 queuing system and the results predicted by the proposed scheme. In the figure, the raw slope is the slope of the modeled CPU utilization, and the predicted slope is the slope of the predicted value of the server workload based on the hybrid depth learning. First, (a) shows a steady pattern of usage rates, but (b) shows a steeply decreasing pattern with a steady utilization rate distribution. (c) is a model that shows a pattern of taking a pole and dropping rapidly. Finally, (d) is a model showing the workload pattern of the IoT application proposed by Bell LAB.

From the experimental results, we can confirm that the prediction slope value is estimated to be almost similar to the CPU utilization distribution slope value. Therefore, if the hybrid deep neural network model predicts the slope of the CPU Utilization like the experimental results, the workload of the server can be predicted. Based on the prediction results, the IoT client virtual machine can efficiently determine the offloading execution.

5. Conclusion

In this paper, we propose the CPU utilization trend, which is one of the workload indicators, through deep learning for efficient offloading execution decision. We use CPU utilization distribution data, which is one of the workload judgment indices, to model workload through deep learning. The currently modeled CPU Utilization pattern reflects the Utilization pattern of the real world, but it shows a simple pattern. Predicted utilization trends are indicators of offloading execution decisions because they represent changes in workload.

In the future, the team will work with the IoT-Cloud fusion virtual machine under study to determine if the IoT virtual machine will decide offloading execution based on the predicted CPU utilization trend. Based on this, we will experiment and analyze how workload estimation of server studied in this paper affects offloading performance.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT and Future Planning (No. 2016R1A2B4008392). And also this work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government (MSIP; Ministry of Science, IC T & Future Planning) (No. 2017R1C1B5018257). This paper was extended from the previous research paper “Performance Estimation Method on IoT-Cloud Environments Using Hybrid Deep Neural Network,” in International Journal of Internet of Things and its Applications, 2018 [14].

References

[1] Y. Son and Y. S. Lee, “Mobile Information Systems”, 2017, **(2017)**.
[2] T. Lin, T. Guo and K. Aberer, “Proceddings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Melbourne, Australia, **(2017)** August 19-25.
[3] Y. S. Lee and Y. Son, International Journal of Smart Home, vol. 8, no. 5, **(2014)**.
[4] K. Yang, S. Ou and H. H. Chen, IEEE Comm. Magazine, vol. 46, no. 1, **(2008)**.
[5] K. Kumar, “Mobile Networks and Applications”, vol. 18, no. 1, **(2013)**.
[6] C. Shi, Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing, Philadelphia, USA, **(2014)** August 11-14.
[7] Y. S. Lee and Y. Son, International Journal of Smart Home, vol. 6, no. 4, **(2012)**.
[8] Y. S. Lee, J. Joeng and Y. Son, Future Generation Computer Systems, vol. 76, **(2017)**.
[9] B. G. Chun, Proceedings of the 6th ACM conference on Computer systems, Salzburg, Austria, **(2011)** April 10-13.
[10] H. T. Dinh, C. Lee, D. Niyato and P. Wang, “Wireless Communications and Mobile Computing”, vol. 13, no. 18, **(2013)**.
[11] H. La and S. Kim, Proceedings of the 7th IEEE International Conference on Service-Oriented Computing and Applications, Matsue, Japan, **(2014)** November 17-19.

[12] M. Schwarz, C. Sauer, H. Daduna, R. Kulik and R. Szekli, "Queueing Systems", vol. 54, **(2006)**.
[13] R. W. Klein and S. D. Roberts, "Simulation", vol. 43, **(1984)**.
[14] Y. Son, S. M. Oh and Y. S. Lee, International Journal of Internet of Things and its Applications, vol. 2, no. 2, **(2018)**.

Authors

Yunsik Son, he received the B.S. degree from the Dept. of Computer Science and Engineering, Dongguk University, Seoul, Korea, in 2004, and M.S. and Ph.D. degrees from the Dept. of Computer Science and Engineering, Dongguk University, Seoul, Korea in 2006 and 2009, respectively. He was a research professor of Det. of Brain and Cognitive Engineering, Korea University, Seoul, Korea from 2015-2017. Currently, he is an Assistant Professor of Dept. of Computer Science and Engineering, Dongguk University, Seoul, Korea. His research areas include secure software, programming languages, compiler construction, mobile/embedded systems, and u-Healthcare.

Seman Oh, he received the B.S. degree from the Seoul National University, Seoul, Korea, in 1977, and M.S. and Ph.D. degrees from the Dept. of Computer Science, Korea Advanced Institute of Science and Technology, Seoul, Korea in 1979 and 1985, respectively. He was a Dean of the Dept. of Computer Science and Engineering, Graduate School, Dongguk University from 1993-1999, a Director of SIGPL in Korea Institute of Information Scientists and Engineers from 2001-2003, a Director of SIGGAME in Korea Information Processing Society from 2004-2005. Currently, he is a Professor of the Dept. of Computer Science and Engineering, Dongguk University, Seoul, Korea. His research areas include smart system solutions, programming languages, and embedded systems.

Yangsun Lee, he received the B.S. degree from the Dept. of Computer Science, Dongguk University, Seoul, Korea, in 1985, and M.S. and Ph.D. degrees from Dept. of Computer Engineering, Dongguk University, Seoul, Korea in 1987 and 2003, respectively. He was a Manager of the Computer Center, Seokyeong University from 1996-2000, a Director of Korea Multimedia Society from 2004-2005, a General Director of Korea Multimedia Society from 2005-2006 and a Vice President of Korea Multimedia Society in 2009. Also, he was a Director of Korea Information Processing Society from 2006-2010 and a President of a Society for the Study of Game at Korea Information Processing Society from 2006-2010. And, he was a Director of Smart Developer Association from 2011-2012. Currently, he is a Professor of Dept. of Computer Engineering, Seokyeong University, Seoul, Korea. His research areas include smart system solutions, programming languages, and embedded systems.

