International Journal of Engineering & Technology, 7 (4.40) (2018) 31-36

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

An Adaptive Offloading Framework for Improving Performance of Applications in IoT Devices Using Fuzzy Multi Criteria Decision Making

Waskitho Wibisono*, Mahaputra Widhi Pande Putu, Tohari Ahmad, Radityo Anggoro

Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia, 60111
**Corresponding author E-mail:waswib@if.its.ac.id*

Abstract

The recent advances of Internet of Things (IoT) technologies have changed the requirements of IoT device to not only provide basic sensing and communication services but also of executing more complex applications with different goals. These challenges have highlighted the need to provide high computation capability in IoT devices. However, common limited resources in IoT devices bring challenges to support application requirements as well as to deal with limited computation resources. To address with this problem, computation offloading can be applied. In this approach heavy computational tasks can be transferred and executed in the cloud computing service to get the result. However, sending heavy computational jobs along with the data to the cloud server are not always efficient, especially where the mobile environments where network performances may changes unpredictably. This paper proposes a prototype of smart offloading framework designed to work in IoT devices using the Fuzzy Multi Criteria Decision Making as the decision tool. The decision whether the job execution will be done in the IoT device itself or being uploaded to the cloud computing server is done by considering internal and external factors such as current network conditions. The smart offloading framework prototype has been developed and tested in a real IoT device. The experiment results showed that the smart offloading approach can improve the performance of applications running in an IoT device by deciding location of job executions in dynamic situations with good results.

Keywords: *Computation Offloading, Cloud Computing, Embedded Systems, Internet of Things, Fuzzy MCDM*

1. Introduction

The proliferations of Internet of Thing (IoT) technologies have enabled IoT devices to become development platforms for various applications. IoT is a concept where devices that are connected to the internet can exchange data and influence devices or physical object in its surrounding [1]. The current advances of IoT devices have enabled various sensors to be embedded in the IoT devices, e.g. GPS, accelerometer, camera, light, gyroscope, proximity sensors etc. There are various systems that use IoT devices as their system components such as in traffic monitoring, disaster mitigation, health monitoring and intelligent building systems etc.
Various applications can be developed and run in IoT Devices. From basic computation applications that perform simple computation to highly computation applications development including augmented reality, image and speech processing etc. [2-4]. These type of applications are generally require heavy computation resources to fulfill the application requirements. On the other hand, IoT devices typically have limited capacity in terms of memory, processing power, network connectivity and energy sources [5]. These facts have highlighted challenges to optimize computation process in the IoT devices.
On the other hand, cloud computing services have provided new era where groups of computation resources such as, networks, servers and various services are easily can be accessed. Various mobile cloud platforms have been deployed for high computation and energy hungry applications that give benefits of reducing energy consumption, preventing overheating and increasing system reliability [5]. These supports can be used to deal with resource limitation of the IoT devices by using the computation offloading technique.
The computation offloading is a technique to execute some tasks of resource constrained devices to the cloud server to speed up the process. It is done by transferring intensive computational process from the IoT devices to more powerful computational resources on the cloud server.
Mobile computation offloading have shown to give benefits in improving performance of the systems deployed in mobile devices [6-8]. Nevertheless this approach need to be executed in an efficient ways to get its benefit. The decision to perform computation offloading can be influenced by various parameters for examples, the current device resources e.g. memory, CPU usage, remaining battery power, network bandwidth etc. All of these factors may influence to the result of mobile computation offloading [9]. Mobile computation offloading have shown to give benefits in improving performance of the systems deployed in mobile devices [6-8]. However problems of computation offloading for heavy computation task such as image recognition done in real IoT devices with limited resources still need to be addressed.
In this paper we present a computation offloading framework to increase the performance of IoT applications by considering the size of jobs to be processed and the current speed network connection in a real time environment. A fuzzy multi criteria decision making has been adopted to deal with the aforementioned challenges and implemented in real IoT devices and cloud computing

platform. Image recognition tasks, i.e. counting number of people in numerous images, has been selected as case study in this paper. A set of extensive experiments to evaluate the performance of the proposed frameworks has been done which show promising results in improving the computation speed in various network traffic conditions.

2. Related Work

Nowadays, various IoT devices with reasonable prices are available on the market. The recent IoT devices are no longer designed for only to provide basic sensing and communication services where currently these devices are designed to have capability of executing more complex applications with different requirements. Example of applications to be executed in IoT devices may range from simple calculation to a very complex applications for image or voice recognition systems [10].

Generally, these types of applications require the devices to have powerful resources for their executions. Remote execution on the cloud server can be used as solution to deal with limitation of IoT devices. Using computation offloading can improve performances of job processing in IoT devices with limited resources such by sending heavy computational jobs to a cloud server [11].

Applying offloading scheme to deal with heavy computation tasks in IoT devices can bring advantages. However various aspects need to be considered to execute this approach including size of tasks, network conditions, server capability and available device resources before making offloading decision. Generally the decision of computation offloading it depends on many parameters such as size of the jobs and network bandwidth.

Various work have been proposed to deal with decision to perform the computational offloading processes. Niu et al. [12] focuses on bandwidth estimation at runtime to design the offloading partitioning models. In [13] Wang et al. focuses on bitrate adjustment at runtime to deal with intermittent connectivity. On the other hand mobile computation offloading have also been applied to increase the performances of smartphone and IoT devices in various applications. Computation offloading approach has been applied to increase performances of smartphone for face detection [14] and speech recognition [15]. In [16], the offloading approach has been used to optimize the process virus scanning.

Ho et al. [17] proposed a mobile data offloading system for video streaming by utilizing multimodal communications (cellular and WIFI links over Software Defined Network (SDN)) by still maintaining quality of the video. The SDN system is responsible to control the traffic by adapting the network conditions. Another work by Zhang et al. [18] proposed a hybrid offloading model to cloudlet queuing model for home automation scenario. A simulation was then developed to evaluate the proposed model.

In this paper we focus on building an adaptive offloading framework in a real IoT device. The main aim of the proposed framework is to improve the performance applications deployed in the device y considering various constraints e.g. number of tasks, network condition and current available resource in the device.

In this paper, we adopt Fuzzy Multi Criteria Decision Making (Fuzzy-MCDM) to deal with computation offloading decision task. This technique brings advantages to deal with uncertainty of decision making given by different criteria and evaluation parameters. For an example if we incorporate different expert assignments toward the importance of each parameter (e.g. network speed, number of tasks, available resources) to create the offloading decision. The approach has been developed as a prototype and deployed in a real IoT device.

The performances of the proposed approach are evaluated in a real environment by using object detection in digital images as a case study. This task utilize deep learning algorithm to recognize available objects in a series of digital images [24,25]. The details of the proposed framework are described in the following sections.

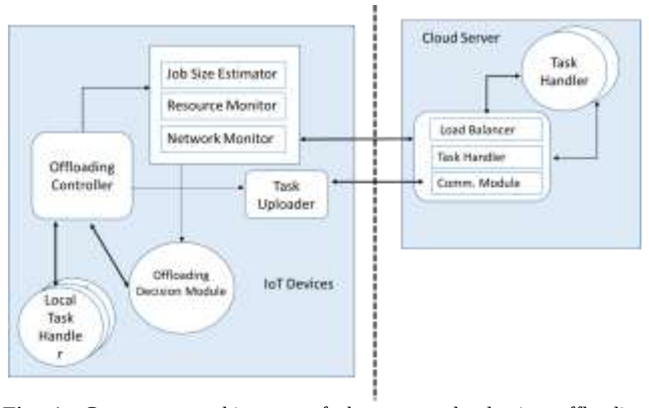

Fig. 1: Component architecture of the proposed adaptive offloading framework to support interaction between IoT devices and cloud server

3. Adaptive computation offloading framework based on fuzzy multi criteria decision making

This paper proposes an adaptive offloading framework to improve performance of applications in IoT devices. To support interoperability and flexibility of the deployment, the framework has been developed using Python programming language since this language is now can be compiled in various IoT devices. The framework is designed to support collaboration between applications in IoT devices and services in the cloud server. The components architecture of the proposed adaptive offloading framework is depicted in Figure 1.

The framework in IoT device side consists of (a) offloading controller who responsible of managing the job processing flow in the device. Using information about the job size, current device resources e.g. available memory, CPU load and network condition (e.g. downloading/uploading speed to/from the cloud) from each load estimator module (b), the offloading decision component (c) will decide whether the current job will be offloaded to the server of being locally executed.

In the cloud side, it consists of (a) incoming task handler, (b) load balancer and (c) communication module. Upon arriving of tasks uploaded by IoT devices, the load balancer component will create associate task handler for each task in the cloud server. Then, the results will be sent back to the corresponding IoT device by the communication module.

3.1. Fuzzy membership functions and fuzzy numbers

One of the important component in the proposed framework at the device side is the offloading decision component. This component is responsible for creating offloading decision given a number of defined criteria. Multi Criteria Decision Making (MCDM) is a decision making tools. It works by selecting best alternative among the available options according to a number of criteria. To deal with this challenge we adopt Fuzzy Multi Criteria Decision Making (Fuzzy-MCDM) as the main decision making tool that utilizes fuzzy set theory in to MCDM process [19].

For each offloading criteria, a set of linguistic variables are defined. In the Fuzzy concept, the linguistic variables represent values in natural or artificial language to represent vagueness of information. The crisp value of each input criteria must be transformed into degrees of membership for each linguistic term. To perform these tasks, the corresponding membership functions need to be defined. The membership function for each criteria represents a degree to which the criteria belong to a set of evaluation to decide offloading execution. For an input criteria qi the membership function using trapezium model $\mu_{trap}(q_i, a, \mathrm{b}, \mathrm{c}, \mathrm{d})$ is defined in the Eq. 1 [20].

$$\mu_{tr}(q_i, a, b, c, d) = \begin{cases} 0 & \text{for } q_i \le a \\ \dfrac{q_i - a}{b - a} & for\ \ a \le q_i \le b \\ 1 & \text{for } b \le q_i \le c \\ \dfrac{d - q_i}{d - c} & for\ \ c \le q_i \le d \\ 0 & \text{for } d \le q_i \end{cases} \qquad (1)$$

Using the trapezium membership function (Eq. 1) the fuzzy number for each linguistic variable of the proposed framework (Low, Medium, High) are constructed as depicted in the following table.

Table 1: Fuzzy Numbers for Linguistic Variables

Linguistic Variables	Fuzzy Numbers
Low (L)	(0.0, 0.3, 0.3, 0.5)
Medium (M)	(0.3, 0.5, 0.5, 0.7)
High (H)	(0.5, 0.7, 0.7, 1.0)

The next step is to assign linguistic variable for each decision criteria used in the offloading framework. There are (1) current available RAM in the corresponding IoT device, (2) current downloading speed from the corresponding cloud server to IoT device, (3) current uploading speed from IoT device to the cloud server, and (4) size of tasks, i.e. the number of images to be processed.
In this paper we use case study of counting number of people in a set of digital images. In this case, the number of digital images need be recognized is the 4th criteria of offloading decision. The membership function for each input criteria for both local and offloading decision are illustrated in the Figure 2.

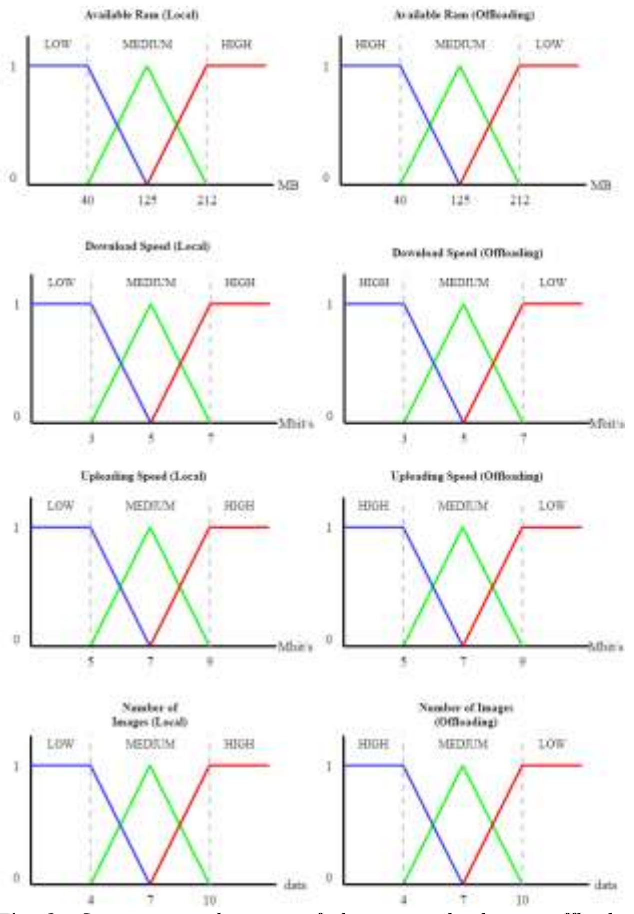

Fig. 2: Component architecture of the proposed adaptive offloading framework to support interaction between IoT devices and cloud server

3.2. Weights assign for each computation offloading criteria

Using the Fuzzy-MCDM concept [21] , evaluation of the importance of each criteria in creating offloading decision (remote execution (offloading) / local execution (without offloading)) are made. In our case, five experts (E1-E5) were surveyed (Step 1) to assign the linguistic variable for each criteria as shown in the following table.

Table 2: Linguistic variable for each criteria assign by 5 experts (L=Low, M= Medium, H= High) .

Criteria	Weights				
	E1	E2	E3	E4	E5
Available Local Ram	M	M	L	M	L
Download Speed	H	M	H	H	M
Upload Speed	M	H	H	M	H
Number of Tasks / Jobs	H	M	H	M	M

In the Step 2, the linguistic variables assigned by the corresponding experts for each criteria are transformed into fuzzy decision matrix as shown in the Table 3.

. **Table 3:** Transformed Fuzzy Decision Matrix

(0.3,0.5,0.5,0.7)	(0.3,0.5,0.5,0.7)	(0.0,0.3,0.3,0.5)	(0.3,0.5,0.5,0.7)	(0.0,0.3,0.3,0.5)
(0.5,0.7,0.7,1.0)	(0.3,0.5,0.5,0.7)	(0.5,0.7,0.7,1.0)	(0.5,0.7,0.7,1.0)	(0.3,0.5,0.5,0.7)
(0.3,0.5,0.5,0.7)	(0.5,07,0.7,1.0)	(0.5,0.7,0.7,1.0)	(0.3,0.5,0.5,0.7)	(0.5,0.7,0.7,1.0)
(0.5,0.7,0.7,1.0)	(0.3,0.5,0.5,0.7)	(0.5,0.7,0.7,1.0)	(0.3,0.5,0.5,0.7)	(0.3,0.5,0.5,0.7)

The average fuzzy numbers are then computed (Step 3) from the defined fuzzy decision matrix. It consisted of all fuzzy number δ_k^i for all criteria C_i where $i = 1,..,y$ assigned by the expert K_k where $k = 1,..,z$. For an example, as shown Table IV, the average fuzzy number of a component a (from trapezium membership function (see Eq. 1) for the available RAM (criteria 1) is obtained as follow $avg_1^a = \frac{0.3+0.3+0.0+0.3+0.0}{5} = 0.18$.
This approach is applied for the remaining fuzzy numbers components of the trapezium membership function for the corresponding criteria i.e. $avg_1^b, avg_1^c, avg_1^d$. The defuzzification value (Step 4) for each criteria d_i is then also computed by averaging these values, i.e. $\phi_i = \frac{avg_1^a+avg_1^b+avg_1^c+avg_1^d}{4}$, e.g. $\phi_i = \frac{0.18+0.42+0.42+0.62}{4} =$ 0.41. Finally (Step 5), the normalized weights for each criteria ϕ_i as shown in Table 4 are obtained using the following equation

$$\phi_i = \frac{d_i}{t(C) * T(\delta_i)} \qquad (2)$$

In the above equation, $t(C)$is the number of criteria and $t(\delta_i)$ denotes the number of fuzzy number assigned for a criteria. The complete results can be seen in Table 4.

. **Table 4:** Normalized Weights for Each Criteria

Criteria c_i	Average Fuzzy Number				(Deff) d_i	*Normalized Weight* ϕ_i
Available RAM	0.18	0.42	0.42	0.62	0.41	0.0256
Downloading Speed	0.42	0.62	0.62	0.88	0.635	0.0397
Uploading Speed	0.42	0.62	0.62	0.88	0.635	0.0397
Number of Tasks	0.38	0.58	0.58	0.82	0.59	0.0369

Using the above tables, fuzzy membership functions numbers, the decision making process is done. For each input criteria and offloading decision, the input value is then fuzzified according to its membership functions as described in the Figure 2. The resulted fuzzy numbers for each criteria is obtained according to the current value of each input criteria exist in the IoT Device (i.e. available RAM at IoT device, downloading speed, uploading speed and number of images to be processed). These fuzzy numbers for each criteria is then averaged to get their defuzzification values. The final decision score of each decision (i.e. local execution or offloading execution) given available input criteria is computed by multiplying the defuzzification values for each criteria with the normalized weight of the corresponding criteria ϕ_i. The total support value for each decision are then obtained and the corresponding actions of the program execution can be performed.

4. Implementation and Evaluation

The proposed smart offloading framework has been developed using python programming language and deployed in Raspberry Pi 2 (B model) equipped with WIFI dongle and connected to an access point. The access point is controlled using a network speed controller software to simulate the dynamic speed of the network during the experiments.

The local execution here is a decision where the jobs are executed in the Raspberry Pi or offloading to cloud server. An example of high computational program to detect number of people in a digital image using deep learning adopted from [22, 23]. The image recognition detection program is deployed both in the cloud server and the Raspberry Pi. This device is equipped with processor ARM Cortex-A7 CPU, Memory 1 GB and a WIFI dongle (See Figure 3). The cloud server is deployed with specification as follow: processor (Intel(R) Xeon(R) CPU E5-2630 @ 2.30GHz, memory 512 MB with Ubuntu 16.04 as the operating system. The illustration of the test bed environment for the experiments is shown in the Figure 3.

In the experiments, all of criteria for offloading decision are obtained directly from the device and network. The available RAM in the device is arranged to be dynamic by creating additional workload that consume its RAM. The bash command *"/proc/meminfo"* is then called for obtaining available RAM in device by the daemon program. The download and upload speed are estimated using speedtest.net library for Python. The set of image for the experiments is copied in a working directory of the program. The number of images need to be processed is then obtained accordingly. Finally all the input criteria are processed using the developed Fuzzy-MCDM offloading decision module and the decision is executed.

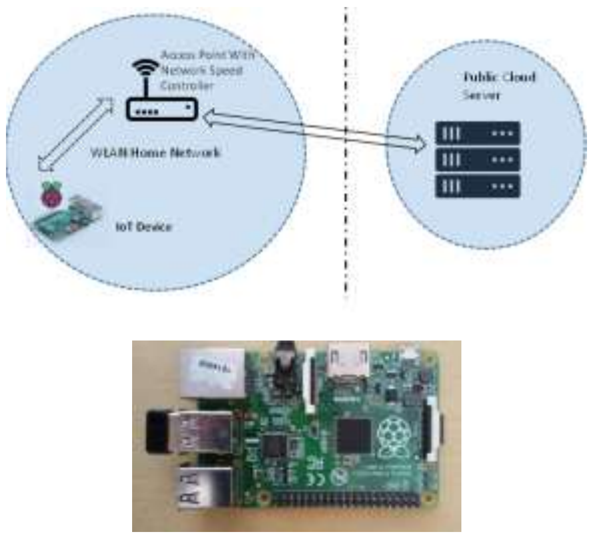

Fig. 3: Test bed architecture (top) and Raspberry Pi 2 (B Model) used in the experiments (bottom)

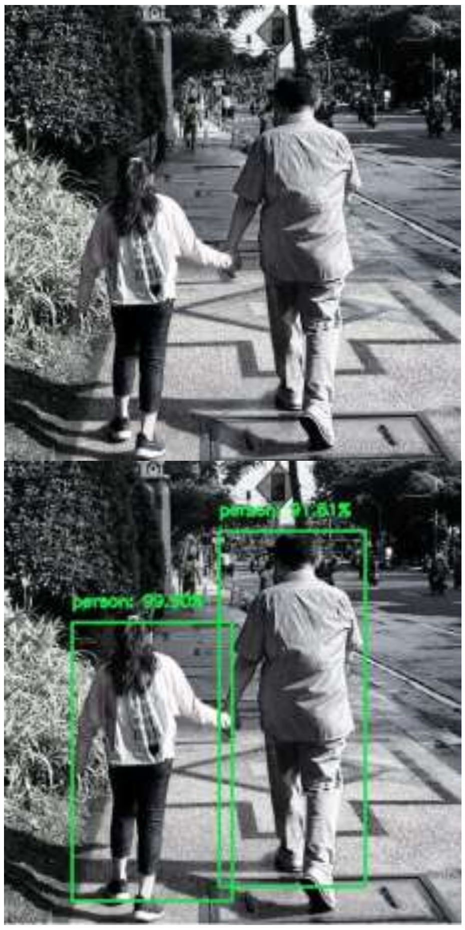

Fig. 4: An example of input image (top) used in the offloading experiment and the resulted images of the running program for offloading scenario (bottom)

3.3. Experiment Scenario

To evaluate the proposed framework , an application to detect number of people in a digital image using deep learning [22, 23] developed in Python programming language is deployed and used to evaluate the adaptive offloading framework. The number of images in each experiments are set to be 4 and 10 images per execution. The experiments for each number of images were conducted 10 times and the average times of computation speed are recorded.

The download (DL) speed and Upload (UL) speed were arranged in the access point device using the network speed controller application. Furthermore, additional workload that consume device RAM in the IoT device was also arranged. The example input and resulted images of the application program for the proposed adaptive offloading scenario are shown in Figure 4.

Table 5 and 6 depict execution time for both local and remote offloading executions using 4 and 10 images per process as the incoming jobs. Each experiment was run twice to obtain time of local and remote (offloading) executions.

The Table 7 and 8 shows the score for each decision option (Local/Offloading) using the Fuzzy-MCDM. These experiments were also computed using the same value of each input criteria in both local and remote offloading experiments. The offloading decision with highest score given the input criteria is selected and the associated times of the selection are taken from Table 5 and 6.

The result in these table show that by using the Fuzzy-MCDM decision tool to select location of the offloading execution location, the total execution time is lower that if all the incoming jobs executed locally in IoT device or remotely executed by uploading the jobs to the cloud server.

Table 5: Execution time for Local and Remote Execution (Number of Images per Process = 4)

No	Available RAM	DL Speed Mbps	UL Speed Mbps	Execution time (s)	
				Local	Offloading
1	169.996	16.8	3.71	89.783	68.38
2	169.748	0.14	3.02	90.116	96.851
3	169.5	0.15	0.23	89.7	113.621
4	54.744	0.14	0.22	90.352	109.602
5	45.496	5.52	2.88	87.335	69.404
6	224.836	0.06	0.24	86.865	108.649
7	223.844	0.16	0.23	86.988	108.49
8	223.72	4.76	0.27	85.887	81.562
9	52.884	6.1	2.06	87.541	67.68
10	152.456	12.37	3.43	88.396	68.407
Total Execution Time				882.963	892.646

Table 6: Execution Time for Local and Remote Execution (Number of Images per Process =10)

	Available RAM	DL Speed Mbps	UL Speed Mbps	Execution time (s)	
No				Local	Offloading
1	69.108	13.94	3.76	202.155	167.37
2	272.02	16.15	3.76	202.821	167.684
3	271.896	5.75	0.27	203.139	212.847
4	267.342	0.15	0.35	201.964	252.43
5	277.394	0.32	0.43	202.376	251.992
6	139.23	5.22	0.42	203.542	212.428
7	57.342	15.23	3.24	203.434	166.472
8	65.772	5.53	0.59	202.942	213.218
9	268.921	6.82	0.45	202.877	212.924
10	127.546	14.39	3.65	202.754	167.23
Total Execution Time (s)				2028	2024.595

Table 7: Decision Made Using Smart Offloading Framework (Number of Images per Process=4)

No	**F-MCDM Score for (Local)**	**F-MCDM Score for (Offloading)**	**Decision Made**	**Execution Time with the Decision**
1	0.51	0.91	Offloading	68.38
2	0.68	0.73	Offloading	96.851
3	0.86	0.56	Local	89.7
4	0.75	0.67	Local	90.352
5	0.48	0.94	Offloading	69.404
6	0.86	0.61	Local	86.865
7	0.86	0.61	Local	86.988
8	0.71	0.76	Offloading	81.562
9	0.48	0.94	Offloading	67.68
10	0.45	0.97	Offloading	68.407
Total Execution Time Using Fuzzy-MCDM				**806.189**

Table 8: Decision Made Using Smart Offloading Framework (Number of Images per Proses = 10)

No	**Fuzzy MCDM Score (Local)**	**Fuzzy MCDM Score (Offloading)**	**Decision**	**Execution Time with the Decision**
1	0.56	0.95	Offloading	167.37
2	0.67	0.89	Offloading	167.684
3	0.94	0.62	Local	203.139
4	0.96	0.45	Local	201.964
5	0.96	0.45	Local	202.376
6	0.94	0.62	Local	203.542
7	0.56	0.95	Offloading	166.472
8	0.86	0.62	Offloading	213.218
9	0.94	0.62	Local	202.877
10	0.71	0.77	Offloading	167.23
Total Execution Time Using Fuzzy-MCDM				**1895.872**

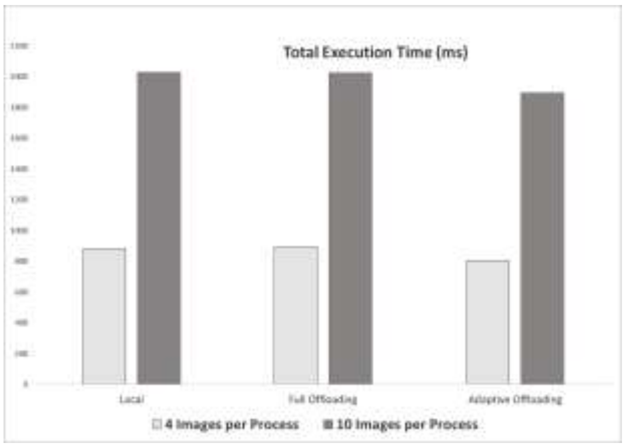

Fig. 4: Summary of total execution times for Local, Full Offloading and the propoped Adaptive Offloading

Results in table VII and Table VIII show that by using the Fuzzy-MCDM decision tool to select the offloading execution location, the total execution time is lower that if all the incoming jobs executed locally in IoT device or remotely executed by uploading the jobs to the cloud server

5. Conclusion

This paper proposed a prototype of smart offloading framework designed to work in IoT devices using the Fuzzy-MCDM as the decision tool. The decision whether the job execution will be done in the IoT device itself or being sent remotely to the cloud server was performed adaptively by considering several criteria. These criteria were related to the capacity of the IoT devices itself as well as external factors such as upload and download speeds between the IoT device and the cloud server.
The proposed framework has been developed to work in real IoT devise where the evaluation of this work was conducted. A program that requires high computational resources was deployed in the device as a case study. The experiments were conducted in dynamic network and device conditions. The results of the experiments showed that the proposed framework can improve the performance of the application running in an IoT device in dealing with location of job execution in dynamic situations. Future work will be focus in dealing with challenges related to energy preservation and dynamic network condition in various mobile computing scenarios.

Acknowledgement

This work was supported by ITS and Ministry of Research Technology and Higher Education (KEMENRISTEKDIKTI) Indonesia, under its competitive research grant, PDUPT Research Grant.

References

[1] K. K. Patel and S. M. Patel, "Internet of Things: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges," International Journal of Engineering Science and Computing,, vol. 6, no. 5, pp. 6122-6132, 2016.

[2] A. Khairi, H. H. Ammar, and R. Bahgat, "Smartphone Energizer: Extending Smartphone's Battery Life with Smart Offloading," in The 9th IEEE Wireless Communications and Mobile Computing Conference (IWCMC), 2013, pp. 329-336,.

[3] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the cloud: Enabling mobile phones as interfaces to cloud applications," in Middleware 2009, 2009, pp. 83–102.

[4] R. Kemp et al., "EyeDentify: Multimedia Cyber Foraging from a Smartphone," presented at the 11th IEEE International Symposium on Multimedia, 2009.

[5] K. Liu, J. Peng, H. Li, X. Zhang, and W. Liu, "Multi-device task offloading with time-constraints for energy efficiency in mobile cloud computing.," Future Generation Computer Systems, vol. 64, 2016.

[6] R. Chandra and P. Bahl, "Maui: Making smartphones last longer with code offload.," in Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services MobiSys ’10, 2010, pp. 49-62.

[7] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in Proceedings of the Sixth Conference on Computer Systems EuroSys ’11, 2011, pp. 301–314.

[8] E. Cuervo et al., "Maui: Making smartphones last longer with code offload," in Mobysys 10, 2010.

[9] A. Bhattacharyaa and P. Dec, "A Survey of Adaptation Techniques in Computation Offloading," Journal of Network and Computer Applications archive, vol. 78, no. C, pp. 97-115 2017.

[10] M. AhmadKhan, "A survey of computation offloading strategies for performance improvement of applications running on mobile devices " Journal of Network and Computer Applications, vol. 56, pp. 28-40, 2015.

[11] Atta ur Rehman Khan, O. Mazliza, A. N. Khan, J. Shuja, and S. Mustafa, "Computation Offloading Cost Estimation in Mobile Cloud Application Models," Wireless Personal Communications, vol. 97, no. 3, pp. 4897–4920, 2017.

[12] J. Niu, W. Song, and M. Atiquzzaman, "Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications," Journal Network Computing Application, vol. 37, pp. 334–347, 2014.

[13] S. Wang and S. Dey, "Rendering adaptation to raddress communication and computation constraints in cloud mobile gaming," in Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM 2010, 2010.

[14] J. Li, Z. Peng, B. Xiao, and Y. Hua, "Make smartphones last a day: Pre-processing based computer vision application offloading," in Proceedings of the 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), 2015, pp. 462–470.

[15] Y. Chang, S. Hung, N. J. Wang, and B. Lin, "Csr: A cloud-assisted speech recognition service for personal mobile device," in Proceedings of the International Conference on Parallel Processing, 2011, pp. 305-314.

[16] W. Zhang, Y. Wen, and Z. Zhang, "Towards virus scanning as a service in mobile cloud computing energy-efficient dispatching policy under n-version protection," IEEE Transactions on Emerging Topics in Computing vol. 6, no. 1, pp. 122-134, 2015.

[17] D. Ho, G. S. Park, and H. Song, "Mobile Data Offloading System for Video Streaming Services over SDN-enabled Wireless Networks " in Proceedings of the 9th ACM Multimedia Systems Conference, 2018, pp. 174-185

[18] J. Zhang et al., "Hybrid computation offloading for smart home automation in mobile cloud computing," Personal and Ubiquitous Computing, vol. 22, no. 1, 2018.

[19] C. Carlsson and R. Fullér, "Fuzzy multiple criteria decision making: Recent developments," Fuzzy Sets and Systems, vol. 78, no. 2, pp. 139-153, 1996.

[20] G. Bojadziev and M. Bojadziev, Fuzzy Logic for Bussiness, Finance and Management (Advances in Fuzzy Systems-Applications and Theory, Vol 12). World Scientific Publishing, 1998.

[21] A. Nagar, "Development of Fuzzy Multi Criteria Decision Making Method for Selection of Optimum Maintenance Alternative," International Journal of Applied Research In Mechanical Engineering, vol. 1, no. 2, 2011.

[22] A. G. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Application," in CoRR, 2017.

[23] A. Rosebrock, "Object detection with deep learning and OpenCV," in https://www.pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/, Accessed May 2018