Dual Threshold Load Balancing in SDN Environment Using Process Migration

Ashis Talukder, Sarder Fakhrul Abedin, Md. Shirajum Munir, and Choong Seon Hong
Department of Computer Science and Engineering, Kyung Hee University, South Korea
Email: {ashis, saab0015, munir, cshong}@khu.ac.kr

***Abstract*—Nowadays, Software Defined Networking (SDN) is a potential evolving technology and considered as an innovative network programming approach. In SDN, energy management and load balancing are considered to be current major research challenges. Previously, these issues are addressed in cloud environments. In this research, we propose a dual threshold load balancing (DTLB) method for SDN environment. The DTLB model works in four steps: process selection, destination physical machine selection, process-physical machine assignment and finally, process migration through docker container. Processes and physical machines are selected based on energy consumption, CPU utilization, and available memory and response time. The effectiveness of the DTLB model is achieved through using the matching technique which assigns a process to an appropriate physical machine and docker container. The docker container is lightweight and easy to develop while providing load balancing, resource sharing, and fault tolerance. The simulation shows that the DTLB model provides energy efficient load balancing with better throughput and faster response time than that of the traditional model without load balancing.**

***Index Terms*—load balancing, SDN, DTLB, physical machine, process migration, docker container.**

I. Introduction

Software Defined Networking (SDN) opens enormous possibilities over the traditional networking by simplifying the management tasks and gains the keen interest of the industry and research community as well [1]. In SDN research, load balancing achieves a key emphasize due to its relevant concerns $e.g.$ energy consumption management, response time improvement, throughput increment, achieving fault tolerance and security.

Previously, many researchers implemented load balancing scheme in cloud environment by employing Virtual Machine (VM) migration [2], [3], [4], while others achieved load balancing by process migration [5], [6]. With the evolution of SDN paradigm, both VM migration and process migration using docker are getting popular in SDN research. Both the techniques are considered to be the potential technologies in improving data center efficiency, such as reducing energy, cost and balancing load [7]. Whatever may be the environment, SDN or cloud environment, the migration of both the VM and docker container have their own beauties but the later one has some salient features over the previous one. Docker container is convenient since it can be executed on any computer, under any framework or infrastructure, and in any SDN or cloud environment, [8] which is not possible in the case of VM migration. A docker image composed of code, runtime, system tools, and system libraries for which it can be run on any machine having docker engine where OS migration is not necessary (see Fig. 1). Unlike VM, the docker-image attaches the complete filesystem that ensures that the process will always run the same as before migration, regardless of the environment. As illustrated in Fig. 1, it is mandatory to migrate the guest operating system (OS) in the case of VM migration. On the other hand, the docker container does not need to include the guest OS in the docker image and thus, no OS migration is necessary. Moreover, VM requires a hypervisor over the infrastructure to run the guest OS. Thus, docker is lightweight and the process migration using docker container incurs lower migration cost, lesser transmit time as compared to VM migration. Moreover, docker is open and provides good security [9]. Therefore, we employ docker container to migrate processes for load balancing in our proposed algorithm.

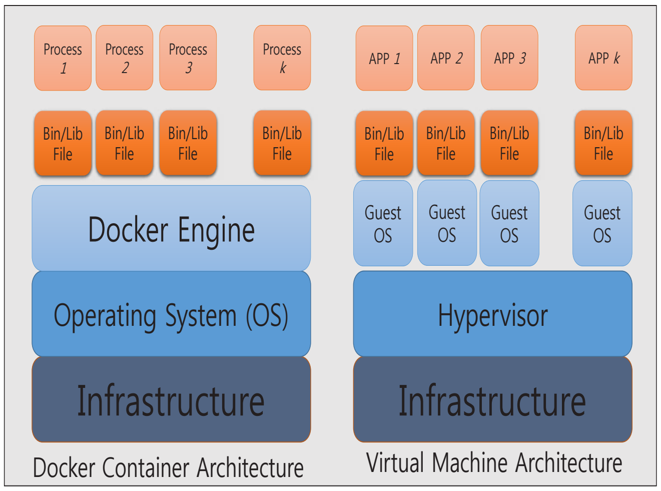

Fig. 1. Docker container and VM Structure.

In this research, we develop a *Dual Threshold Load Balancing (DTLM)* model for SDN based enterprise environment. The DTLM module works in the SDN controller to manage the servers, physical machine, and processes running on them. It collects system information and compares with two threshold values to check the overload and underload conditions. The LBM module migrates processes using docker container to appropriate physical machine determined by one side match-

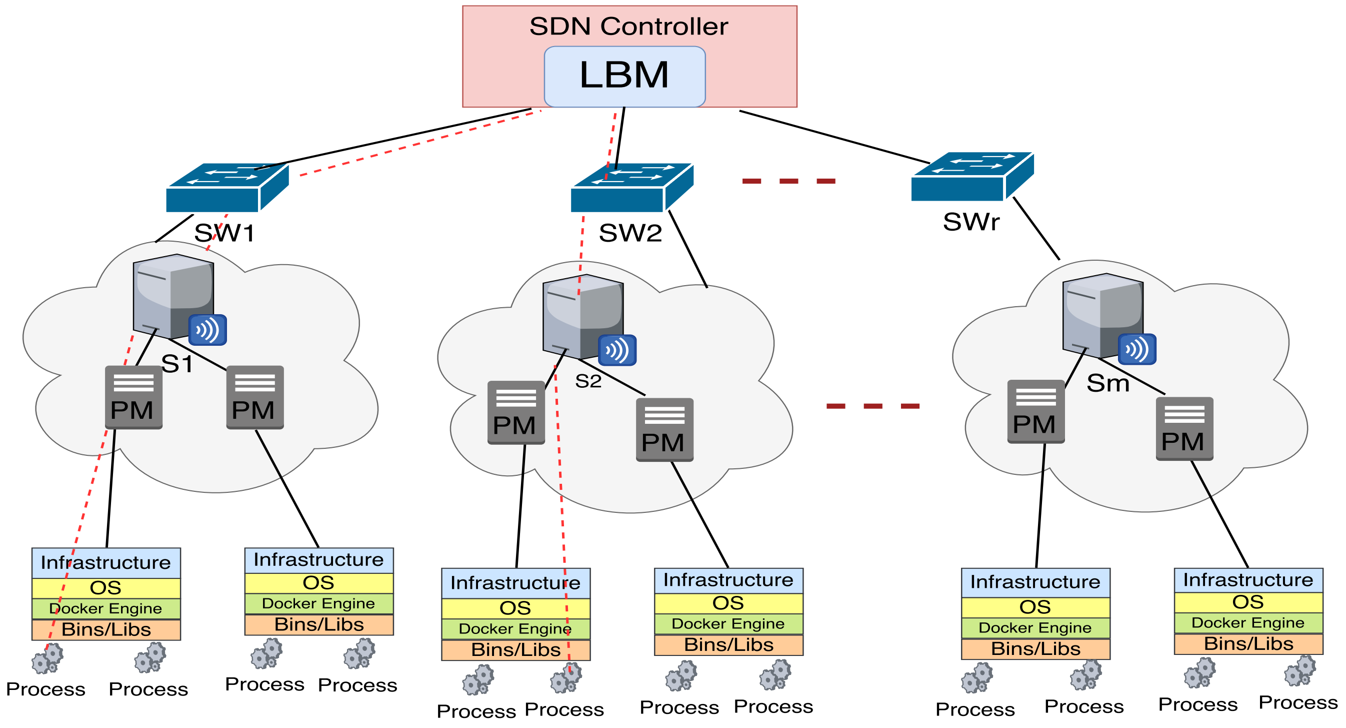

Fig. 2. SDN system model for DTLB model.

ing. The DTLM possesses the feature of dynamic load balancing and efficient resource management, reducing energy and memory consumption, increasing throughput measured by the number of process completion in unit time, and portability. The DTMI leverages the matching technique and docker container as well.

The organization of the rest parts of the paper has the following order. In section II, we describe the state-of-the-art of load balancing techniques. We provide a system model in section III. In section IV, the proposed Dual Threshold Load Balancing (DTLB) model is stated. The performance evaluation of the proposed model is presented in the section V and the conclusion in section VI.

II. Existing Study

VM migration is very common in a cloud environment and many researchers have conducted research on could based VM migration in huge extent in recent years. With the evolution of SDN, researchers also employ VM migration in SDN environment. The technique of process migration in SDN is also getting popular for its various salient features.

Wang et al. [10] determine the optimal migration sequence and transmission rate subject to minimize the total migration time. In our research, we employ the matching strategy to find the best assignment of a process to an appropriate physical machine that reduces the migrations frequency and eventually enhances the performance. Zhang et al. [11] propose a heuristic VM migration method which resumes the network connection very fast during migration in SDN environment. This feature facilitates fast live migration.

Namal et al. [12] develop a load balancing technique for SDN system facilitated by flow admission control. Their model reduces the number of unsatisfied users as well as improves resource allocation.

Adhikari et al. [13] implement a VM migration based, energy-aware load balancing method which uses two thresholds values for migration decision making. However, their technique does not work in the case of a local cloud. Meanwhile, our proposed DTLB model is developed for local SDN system which is applicable in geographically distributed systems.

The researchers in [2] formulate a VM allocation policy for managing resources in cloud data centers which ensures energy efficiency and reliability quality as well. An energy efficient load balancing technique using VM migrated is developed in [3] for virtualized data centers. It ensures high resource utilization and minimizes the number of migrations as well.

A threshold based dynamic load balancing technique for heterogeneous cloud environment is developed by Mishra et al. [4]. The method maximizes resource utilization as well as provides energy efficiency.

Most of the researchers mainly focus on reducing energy consumption and load balancing. Our DTLB algorithm reduces energy consumption and response time with the increased amount of throughput, efficient memory management, and load balancing.

III. System Model

Let us consider an SDN based system environment powered by OpenFlow protocol as illustrated in Fig. 2, where there are many switches. Each of the switches is connected to an SDN controller. The switches are denoted by $SW = \{SW_1, SW_2, ..., SW_r\}$. Each switch is attached with a number of servers $S = \{S_1, S_2, ..., S_m\}$. Each server has multiple instances of physical machines, $PM = \{PM_1, PM_2, ..., PM_n\}$ and memory units $M = \{M_1, M_2, ..., M_n\}$ linked with it. Here, the total memory size is $M_{size} = \sum_{i=1}^{n} M_i$. Users submit their processes timely when it is necessary and the processess are served by the servers. Processes are assigned with the physical machines. Each physical machine serves a set of processesses $P_i = \{P_{i1}, P_{i2}, ..., P_{ik}\}$ on its associated memory unit M_i. Each process P_i has its memory requirement m_i.

We will design a dual threshold algorithm to balance the load of the physical machines $i.e.$ servers by migrating process to an appropriate server.

IV. Dual threshold Load balancing (dtlb) Model

In this section, we design the proposed process migration algorithm in the SDN environment. We design a Load Balancing Module (LBM) that with work in the system's SDN controller as illustrated in the Fig. 3. The algorithm (Alg. 1) is a continuous process and executed in four steps: Process (P) selection, Physical machine (PM) selection, Process-physical machine (P-PM) assignment and Process migration using docker container. We consider various parameters such as memory requirement, available memory size, CPU utilization (indicated by the load of the physical machines) and, the response time of the physical machine to accomplish previous four steps discussed below.

It is possible to estimate the current load or CPU utilization of a physical machine, $E(PM_i)$ by its power consumption. We employ the findings of the researchers $e.g.$ [7], [14], [15], [16], where they have found that the CPU utilization is linearly related to power consumption. Their research also suffices that seventy percent power is necessary to keep an idle server operation. Thus, the power consumption over 70% contributes to the server load or CPU utilization of the physical machine PM_i and is given by:

$$e(PM_i) = E_i - \eta E_{max} \tag{1}$$

where, $E_{max} = 250W$ for standard server [7], [4], E_i is the current energy consumtion of the PM_i, and $\eta = 0.70$ [7] [13].

We normalize the load of a PM estimated in the Eq. 1 to the range $(0, 1)$ using *min-max normalization* [17] as:

$$E(PM_i) = \frac{E_i - \eta E_{max}}{(1 - \eta) E_{max}} \tag{2}$$

The available or residual memory of a PM_i is determined by the following equation:

$$RM_i = M_i - \sum_{j=0}^{k} m_j \tag{3}$$

where, $\sum_{j=0}^{k} m_j$ is the currently used memory by k processes running on PM_i.

Another parameter, response time (T_i) measures the load of the physical machine. The higher value of T_i indicates that the PM_i is more overloaded. The response time of a physical machine, PM_i is given by:

$$T_i = t_w + t_e \tag{4}$$

where, $t_w =$ is the waiting time, $t_e =$ is the actual executing time. The waiting time, t_w, is the queuing time and can be taken as the average of all the processes submitted to a physical machine. The execution time, t_e, is determined by dividing the process size by PM's CPU clock speed (BIPS or MIPS).

The term, propagation time, t_p, might be included when the PMs are placed in different geographical areas and can be computed by dividing the process container size (MB) by the link bandwidth (Mb/s). In our model, we have ignored the propagation time since the PMs are placed nearby in our system model.

A. Process (P) Selection

In this step, a candidate list of processes is selected for migration from overloaded and/or underloaded physical machines (lines 2 - 10 in Alg. 1). The current load, $L(PM_i)$, of a physical machine, PM_i, depends on parameters such as energy consumption (or CPU utilization) and response time combinedly and is formulated as:

$$L(PM_i) = \alpha E(PM_i) + (1 - \alpha) T_i \tag{5}$$

where, $\alpha \in (0, 1)$, is priority parameter between $E(PM_i)$ and T_i. A higher value of both the energy consumption and response time indicates the load of the PM is higher.

The conditions for a PM_i of being either overloaded or underloaded are determined by two thresholds $(\theta_{min}, \theta_{max})$ and this suffices the name *Dual Threshold Load Balancing (DTLB)* algorithm. If $L(PM_i) < \theta_{min}$, the PM_i is considered to be underloaded and all the processes running on it are migrated to other physical machines and the PM_i is set to sleeping state. On the other hand, if $L(PM_i) \geq \theta_{max}$, the PM_i is overloaded and the processes with higher process ID are selected for migration. Here, higher process ID means the processes that have been started execution most recently.

B. Physical Machine (PM) Selection

A list of destined physical machines where the selected processes could be migrated is selected depending upon the available or residual memory of the server and response time (lines 14 - 20 in Alg. 1). The PM to be selected for new process migrated to it, it must be lightly or moderately loaded which is determined by the value $L(PM_i)$ compared with the thresholds. Furthermore, it must have enough available or residual memory as determined by Eq. 3.

C. Prcess-Physical Machine (P-PM) Assignmnet

After selecting the processes for migration and the destination physical machines to migrate those processes, now we find an appropriate process-physical Machine (P-PM) assignment by one-sided matching [18] from processor side (lines 18 - 19 in Alg. 1). This is done to assign a process to a proper physical machine. A preference list for processes' side of selecting physical machine is created according to the requirements of the process execution on the physical machine. Requirements include residual memory, response time, power consumption.

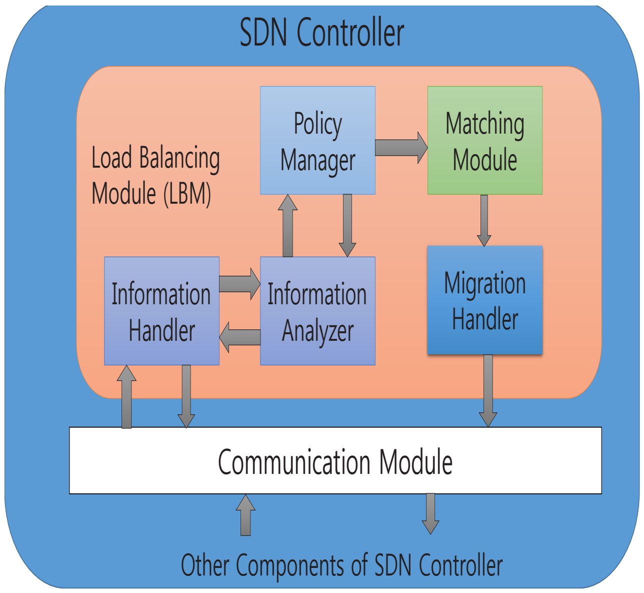

Fig. 3. Load balancing module (LBM).

D. Process Migration

Finally, the selected processes are migrated to the desired physical machines according to the to $P-PM$ assignment accomplished by matching technique in the previous step. Now, the process migration is carried out using docker container (lines 21 - 25 in Alg. 1). If there is any case of migrating all the processes of any underloaded PM, the PM is set to be idle or sleeping state.

The migration process is monitored and accomplished by the Load Balancing Module (LBM) as depicted in the Fig. 3, by working with the SDN controller. The Information Handler collects the necessary information via Communication Module from SDN controller. With this information, the Information Analyzer module determines the overloaded and underloaded status of the physical machines. Then, the Policy manager selects the candidate processes and physical machines. The Matching Module performs the P-PM assignment and finally, the Migration Handler migrates the processes with the intervention of SDN controller.

V. Performance Evaluation

We have performed a simulation with synthesized data by executing python programmes. We compare the energy con-

Algorithm 1 Dual Threshold Load Balancing (DTLB) Module

```
1: Initialize $PMs$ with some randomly created and assigned
   processes $P_{ij}s$ to them
2: **Repeat**{
3: Get parameter information $E(PM_i), T_i, RM_i, L(PM_i)$
   and determine which $PMs$ need process migration com-
   paring with a thresholds $\theta_{min}$ and $\theta_{max}$.
4: **if** (No migration is necessary) **then**
5:     Go to step 3 after a predefined time quantum
6: **end if**
7: **if** (any $PM_i$ has $L(PM_i) <$ than $\theta_{min}$) **then**
8:     List all $P_{ij}s$ with higher process ID from the $PM_i$
9:     Tag the $PM_i$
10: **end if**
11: **if** (any $PM_i$ has $L(PM_i) \geq \theta_{max}$) **then**
12:     List all the processes $P_{ij}s$ with higher ID from the
        $PM_is$
13: **end if**
14: Collect residual memory capacity $RM_i$ of the $PM_is$ to
    select where to migrate and compare two parameters
    energy consumption and residual memory capacity of
    $PM_i$
15: **if** (No migration is necessary) **then**
16:     Go to step 3 after a predefined time quantum
17: **else**
18:     Make preferences list of $PMs$ for $P_{ij}$ based on the
        energy and memory requirements
19:     Calculate the $P_{ij}$ - $PM_k$ stable matching
20: **end if**
21: **FOR** (each $P_{ij}$ - $PM_k$ pair)
22: Migrate the $P_{ij}$ to the destination $PM_k$ using Docker
    container
23: **if** there is any $PM_i$ with Tag **then**
24:     Free the $PM_i$ to idle state
25: **end if**
26: **EndFor**
27: }**until(False)**
```

sumption, memory usages, and throughput with the traditional model without load balancing techniques.

A. Simulation setup

We perform simulation using Python with synthesized data on a machine with features: Core $i3$, 3.90 GHz processor, and 8 GB RAM. We create a list of 100 processes, each having physical size randomly taken from $[500, 750, 1000]$ MB, containing number of instructions randomly chosen from $[500000, 1000000, 5000000]$, and having energy requirement selected from $[3, 5, 7, 10, 20, 25, 30]$ Watts.

We use 10 physical machines, each having $E_{max} = 250W$ [13], [7], physical memory $m = 2$ GB, CPU clock speed is randomly picked up from $[1000, 2000, 25000]$ MIPS. Generally, a physical machine is 100% utilised if its power consumption is $E_i = 250$ W and an idle physical machine consumes 70% of the maximum energy, E_{max}.

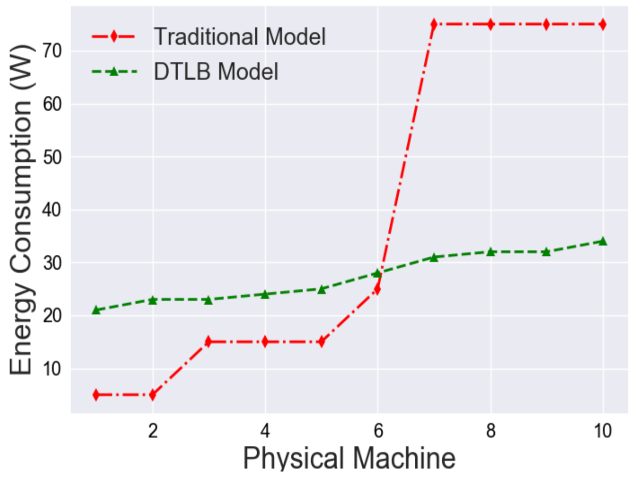

Fig. 4. Energy Consumption.

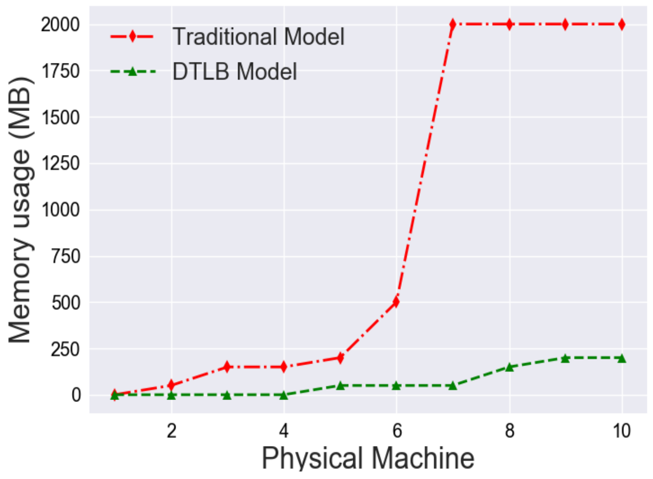

Fig. 5. Memory usage.

We assume that the threshold value $\theta_{min} = 0.25$ and $\theta_{max} = 0.80$ for our experimets. The process arival time is generated by Poisson distribution with $\lambda = 25ms$.

B. Result Analysis

In order to perform the result analysis, we compare the result of our DTLB model with that of the Traditional model without load balancing. Here, we consider three parameters for performing analysis $e.g.$ energy consumption (CPU utilization), memory usage, and throughput.

1) Energy consumption: We have measured the CPU utilization of physical machine, PM_i, in terms of energy consumption. The Fig. 4 depicts that the DTLB model outperforms the traditional model with the increased number of physical machines and process arrival with time. That means our algorithm ensures the elevated CPU utilization with the increase of physical machines and processes over time.

2) Memory usage: The memory usage is related to energy consumption and response time as well. The lower the memory usage, the lesser the power consumption and the better the response time. The experimental result shows that our DTLB model utilizes about four times less amount of memory as compared to the traditional model as illustrated in the Fig. 5. Less memory utilization means lower transmission time which would be convenient if the physical machines are placed geographically apart. Thus, our model is scalable to long distanced system architecture as well.

3) Throughput: We measure the throughput in terms of the number of process completion over time. Here, throughput is also the measurement of load balancing and better response time indirectly. So the number of completed processes indicates better load balancing and attractive response time. The Fig. 6 also depicts that our algorithm provides almost two times better throughput than that of the traditional model.

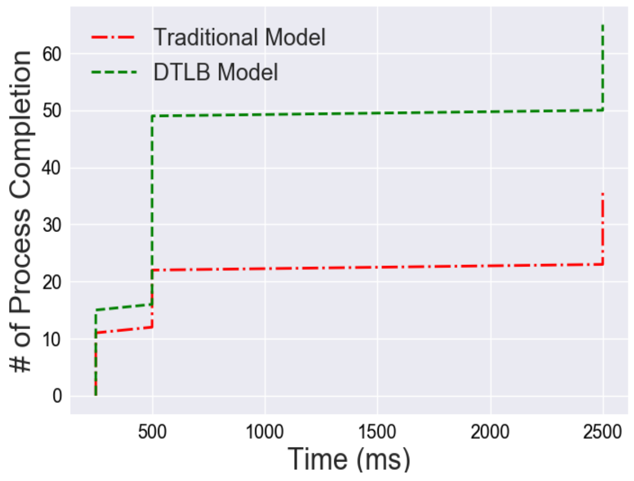

Fig. 6. Throughput (Number of process execution over time).

VI. Conclusion

In this paper, we propose a Dual Threshold Load Balancing (DTLB) model for SDN based enterprise system. The DTLB model selects processes from overloaded and underloaded physical machines and migrates to some appropriate physical machines. In order to select the best physical machine to migrate a process, it applies the popular matching strategy. Finally, it migrates the processes using docker containers. The DTLB provides the best P-PM assignment by matching and ensures less transmission cost, energy consumption, memory usage while improving throughput and load balancing. It also provides portability, fault tolerance, and security for using docker container.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2017R1A2A2A05000995).
*Dr. CS Hong is the corresponding author.

References

[1] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, "Adaptive resource management and control in software defined networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 18–33, 2015.

[2] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proceedings of the 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing*. IEEE Computer Society, 2010, pp. 826–831.

[3] H. Maziku and S. Shetty, "Network aware vm migration in cloud data centers," in *Research and Educational Experiment Workshop (GREE), 2014 Third GENI*. IEEE, 2014, pp. 25–28.

[4] S. K. Mishra, M. A. Khan, B. Sahoo, D. Puthal, M. S. Obaidat, and K. Hsiao, "Time efficient dynamic threshold-based load balancing technique for cloud computing," in *Computer, Information and Telecommunication Systems (CITS), 2017 International Conference on*. IEEE, 2017, pp. 161–165.

[5] C. A. Scheurer, H. K. Scheurer, and P. G. Kropf, "Load balancing driven process migration," Tech. rep., Inst, Tech. Rep., 1995.

[6] D. W. Glazer and C. Tropper, "On process migration and load balancing in time warp," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 3, pp. 318–327, 1993.

[7] A. Kella and G. Belalem, "Vm live migration algorithm based on stable matching model to improve energy consumption and quality of service." in *CLOSER*, 2014, pp. 118–128.

[8] "Docker Documentation: docker docs," https://docs.docker.com/, accessed: 2017-09-20.

[9] A. Talukder, S. F. Abedin, A. K. Bairagi, M. G. R. Alam, S. Kim, S. Lee, and C. S. Hong, "Process migration using docker container in sdn environment," *Korean Computer Congress (KCC)*, pp. 1331–1333, 2016.

[10] H. Wang, Y. Li, Y. Zhang, and D. Jin, "Virtual machine migration planning in software-defined networks," *IEEE Transactions on Cloud Computing*, 2017.

[11] S. Q. Zhang, P. Yasrebi, A. Tizghadam, H. Bannazadeh, and A. Leon-Garcia, "Fast network flow resumption for live virtual machine migration on sdn," in *Network Protocols (ICNP), 2015 IEEE 23rd International Conference on*. IEEE, 2015, pp. 446–452.

[12] S. Namal, I. Ahmad, A. Gurtov, and M. Ylianttila, "Sdn based inter-technology load balancing leveraged by flow admission control," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*. IEEE, 2013, pp. 1–5.

[13] J. Adhikari and S. Patil, "Double threshold energy aware load balancing in cloud computing," in *Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on*. IEEE, 2013, pp. 1–6.

[14] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2. ACM, 2007, pp. 13–23.

[15] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster computing*, vol. 12, no. 1, pp. 1–15, 2009.

[16] R. Sinha, N. Purohit, and H. Diwanji, "Energy efficient dynamic integration of thresholds for migration at cloud data centers," *IJCA Special Issue on Communication and Networks*, vol. 1, pp. 44–49, 2011.

[17] L. Al Shalabi, Z. Shaaban, and B. Kasasbeh, "Data mining: A preprocessing engine," *Journal of Computer Science*, vol. 2, no. 9, pp. 735–739, 2006.

[18] S. F. Abedin, M. G. R. Alam, N. H. Tran, and C. S. Hong, "A fog based system model for cooperative iot node pairing using matching theory," in *Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific*. IEEE, 2015, pp. 309–314.