Proc. Of the International E-Conference on Advances in Engineering, Technology and Management - ICETM 2020

ISBN: 978-1-63248-188-7 DOI:10.15224/978-1-63248-188-7-04

Virtual Machine Management Technique for Providing High-Efficiency DaaS to Companies

Baikjun Choi, Changsung Lee, Sooyong Park

***Abstract*—The provision of Desktop as a Service (DaaS) to companies using virtualization technology comes with advantages that improve service quality and reduce the costs incurred from service provision and maintenance. These advantages originate from the efficient use of resources in physical servers through virtual machine management. A virtual machine has various resource allocation and usage options in a single physical server where the machine is operated. Thus, a virtual machine management technique, such as virtual machine allocation and placement based on demand, enables the efficient management of data resources in the machine in accordance with the use patterns of users, even under complex load characteristics. Capitalizing on these innovative affordances, this study developed a virtual machine management technique for the efficient management of physical server resources. Companies can confirm the efficiency of the proposed technique via simulations grounded in resource use data from virtual and physical servers in a DaaS environment.**

***Keywords*—DaaS., Virtual Machine, Load Balancing, Cloud, VDI**

I. Introduction

The virtual desktop infrastructure (VDI) is a technology that enables the operation of virtualized desktops in a user's local terminal. A VDI produces output that is then stored in a VDI operation server located in remote data centers rather than in client storage devices [1]. Correspondingly, all tasks that are performed by users after connecting to a virtualized desktop through user clients are executed and stored in the operation server [1]. These tasks include the operation of programs and applications, the execution of processes, and the use of data, including those on operating systems [1].

A virtualized desktop is delivered to users through the cloud computing-based, outsourcing business called Desktop as a service (DaaS) [2]. In this process, cloud service providers create virtualized desktops or virtual machines (VM) in a VDI operating server and allocate them to DaaS users. Users then employ the assigned virtual machines through the Internet (cloud) and can use the information stored in these machines anytime, anywhere because such information is kept in the VDI operating server installed in a specific data center.

DaaS can be classified on the basis of user type and supply target, that is, DaaS for personal users and DaaS for companies.

Users who avail of a VDI in a DaaS environment use the service by allocating the virtual machines created in a data center's VDI operating server after executing the steps shown in Figure 1. When a user logs in and requests virtual desktop services by running programs for connection from a user terminal, user authentication is implemented in the VDI management server, after which a virtual machine is created and placed in a specific physical server or physical machine (PM) out of the VDI operating server. The operating virtual machine is then assigned to the user, and this machine transfers screen information to the user's terminal in video form.

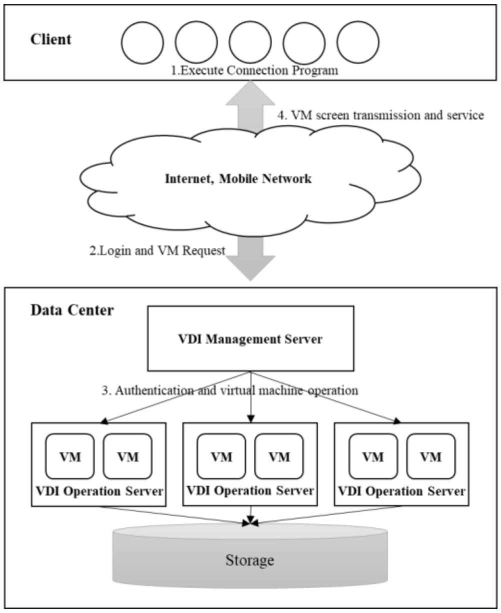


Figure 1. DaaS execution scenario using VDI

The usage patterns and workloads of the virtual machines run for companies as DaaS users are characterized by standardized use patterns that arise from the business purposes specific to these enterprises. By analyzing and using these patterns, the resource requirements of virtual machines and the resource utilization of a physical server can be deterministically predicted.

In consideration of the above-mentioned issues, this study established a method for efficiently providing DaaS to companies. The principle underlying the method is the minimization of performance degradation due to overload in specific physical servers by means of load distribution in relation to resource utilization in a physical server. This distribution is carried out in accordance with the resource requirements of virtual machines after their allocation and placement are determined by deploying the proposed virtual

Baikjun Choi, Changsung Lee, Sooyong Park

Department of Computer Science and Engineering
Sogang University
Seoul, Republic of Korea

machine management technique in a VDI management server. The technique is activated once users log in and request virtual machines and services, with the above-mentioned processes executed in a DaaS environment.

II. Related Literature

A type of load balancing algorithms is Sandpiper [3], which calculates and determines the migration timing of virtual machines for load balancing in physical servers; migration is performed using black box and gray box approaches. Sandpiper has monitoring and profiling engines that generate profiles by analyzing how real-time resource utilization information (CPU [central processing unit] usage, memory usage, network usage, etc.) is monitored in virtual machines and physical servers. It detects resource shortage by keeping track of resources in virtual machines and physical servers and determines overloading in physical servers by setting an arbitrary threshold value. If this threshold is exceeded during resource utilization in a physical server, the server is considered to be in an overload state, thereby triggering the need to perform load balancing in the system. If additional resources cannot be allocated to virtual machines or additional allocation cannot be carried out to resolve overload, virtual machines are migrated.

The resource utilization rate is defined as a represented value called volume, which refers to the utilization rate of multiple resources. In this regard, load balancing is executed by an algorithm on the basis of volume thus:

$$Volume = \frac{1}{1-CPU} \times \frac{1}{1-Mem} \times \frac{1}{1-Net} \qquad (1)$$

where, cpu, mem, and net refer to the resource utilization rates of physical servers and virtual machines that reduce complexity through load balancing based on volume. However, because each individual resource load embodied in volume is disregarded, even under volume-driven load balancing, the maximum value of the resources encompassed by the scope of volume can be duplicated, thus potentially causing overload in a physical server. This problem, in turn, may induce the unnecessary migration of virtual machines.

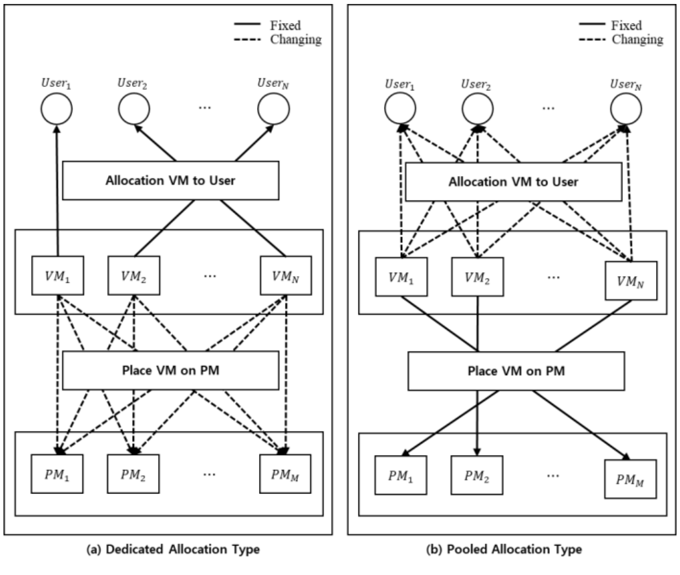
Algorithms other than Sandpiper have been studied [3], including the hybrid genetic-based host load aware algorithm (HGBLA) [4]. HGBLA verifies physical server loads and user constraints with a heuristic approach when initially carrying out virtual machine placement. Researchers proposed method of optimizing the placement of virtual machines using the fitness function-based hybrid genetic algorithm [4]. They also put forward a technique for balancing load in a physical server, in which the dynamic allocation of virtual machines is adjusted and resource utilization in a physical server is modified. In [5], the researchers proposed a method called Distributed load balancing algorithm based on compare and balance (DLABA-CAB), which performs adaptable real-time migration of virtual machines by calculating migration costs on the basis of log data on resource utilization through the lblog program, which monitors resource utilization in a physical server to balance load in a cloud system [6].

Despite the various methods and algorithms proposed in [3], [4], and [5] to resolve load balancing problem, these did not look into the usage patterns and workloads of virtual machines employed by companies that avail of DaaS. These approaches therefore incur unnecessary scheduling overhead and suffer from considerable delays. That is, the methods implement unnecessary work, such as the migration of virtual machines in a time slot wherein resource utilization rapidly increases (e.g., the onset of work) if the resource requirements of virtual machines and the usage patterns of users are neglected. Ultimately, these deficiencies cause tremendous delays and problems that arise from host overload. Because most users perform ordinary routine work every day, the resources needed by virtual machines in this respect can be deterministically predicted to some extent. Correspondingly, services can be efficiently provided through a virtual machine management technique that is anchored in the analysis of usage pattern and resource utilization.

III. Proposed virtual machine management technique

A. *Allocation of virtual machines*

Methods for virtual machine allocation, as implemented in the technique proposed in this work, are categorized into dedicated and pooled allocation—a classification based on the relationship among users, virtual machines, and physical servers in terms of allocation and placement. Figure 2 illustrates these allocation approaches.



(a) Dedicated Allocation Type

(b) Pooled Allocation Type

Figure 2. Methods of allocation virtual machines

Dedicated allocation involves the creation and assignment of M virtual machines by a user (Figure 2(a)), and the allocation relationship between a user and a virtual machine is fixed to 1:M. The virtual machine is exclusively dedicated to the allocated user, which can always employ the same virtual machine as its own desktop environment. In this method, as well, the allocation relationship between user and virtual machine is fixed (fixed allocation). Thus, the load borne by the physical server is balanced given the change in virtual machine placement (placement change) performed in the server.

In pooled allocation, users and virtual machines are in an N:M multi-allocation relationship (Figure 2(b)), in which N users share M virtual machines via the connection of M virtual machines to a single pool of virtual machines. When a user logs in, one of the virtual machines in the pool is allocated to the user, and when the user logs out, this machine is returned to the group. Because the placement relationship between physical server and virtual machine is fixed (fixed placement), the load shouldered by the former is balanced given changes to allocation information (allocation change) in the virtual machine with regard to the user. Table 1 summarizes the comparison of the dedicated and pooled allocation methods.

TABLE I. COMPARISON OF DEDICATED AND POOLED ALLOCATION METHODS

Item	Allocation Methods	
	Dedicated	***Pooled***
Allcoation relationship (user:virtual machine)	1:M	N:M
Virutal machine allocation	Fixed	Changing
Virtual machine placement	Changing	Fixed

A virtual machine is allocated on the basis of the characteristics of how a user employs a virtual machine. Such attributes are determined in accordance with the operating objective of a VDI and the policy that governs DaaS provision to companies. The operating objective can be divided into VDI for business and VDI for the Internet. In the former, the dedicated allocation method is applied because the VDI employs applications for business in a virtual machine, which therefore requires constant connection when a user requests for a virtual machine and all the data in this machine needs to be preserved. In contrast, because VDI for the Internet is characterized by a low rate of work execution in a virtual machine (30% to guarantee the number of concurrent connection users) and because the periods for connection maintenance and connection attempts are often irregular, the pooled allocation method is employed within the range that guarantees the number of concurrent connection users necessary to reduce VDI construction costs [7].

The algorithm put forward in this work monitors and collects the resource utilization data of physical servers and virtual machines. The collected data serve as bases for the analysis of the use patterns and demands for resource utilization of users. Subsequently, the algorithm optimizes the static allocation of virtual machines to users on the grounds of the analysis results and the placement of virtual machines in physical servers, with a view to achieving load balance in the system. The algorithm can prevent overload occurrence in a host and delays in virtualization service for users, which stem from re-placement and dynamic migration. This preventive measure is made possible by the load balancing optimization coursed through the static allocation and placement of virtual machines to users. This allocation is founded on an examination of usage patterns and the prediction of resource utilization in physical servers and virtual machines.

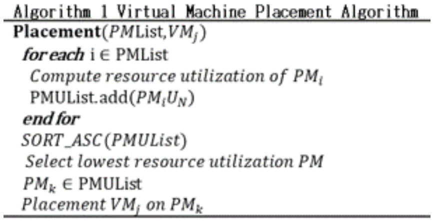
B. ***Virtual machine placement algorithm***

The virtual machine allocation and placement algorithm proposed in this study can be deployed in accordance with the virtual machine allocation method. Given that the allocation relationship between user and virtual machine is fixed to 1:1 in dedicated allocation, the assignment of a virtual machine to a user does not change; such an alteration is applicable only to the placement of virtual machines in physical servers. Thus, the load imposed on a physical server is balanced by activating the virtual machine placement algorithm, which alters the placement of virtual machines in a server.

Once the placement of virtual machines is requested because of the creation of such machines, 1) prediction data of resource utilization (Physical Machine Utilization or PMU) are calculated for each physical server by evaluating the log data of resource utilization in a single server against the data of all the physical servers in a data center (1).

$$PM_i U(N) = \frac{N-1}{N} PM_i U(N-1) + \frac{1}{N} PM_i U_N \tag{2}$$

Such data are generated for all physical servers, after which 2) the resource utilization data of each physical server are stored in a PMUList. Once prediction data are generated for all physical servers, the server whose resource utilization is the lowest, as reflected in the PMUList, is selected for the placement of a virtual machine. Figure 3 shows the pseudocode executed by the virtual machine placement algorithm under dedicated allocation.



```
Algorithm 1 Virtual Machine Placement Algorithm
Placement($PM$List,$VM_j$)
  for each i ∈ PMList
    Compute resource utilization of $PM_i$
    PMUList.add($PM_iU_N$)
  end for
  SORT_ASC($PMUList$)
  Select lowest resource utilization PM
  $PM_k$ ∈ PMUList
  Placement $VM_j$ on $PM_k$
```

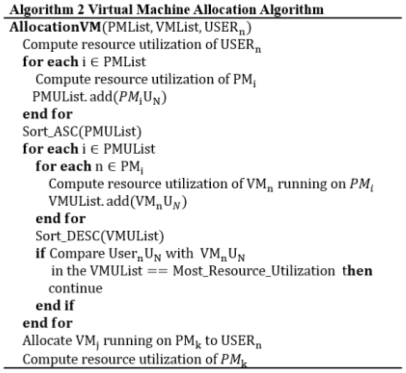
Figure 3. Pseudocode of the virtual machine placement algorithm

C. ***Virtual machine allocation algorithm***

When pooled allocation is used, virtual machines are not migrated or re-placed, but the virtual machine allocation algorithm that balances load in a physical server is activated by changing the allocation information of a virtual machine in relation to a user. The pooled allocation method assumes that virtual machine placement in a physical server is complete. Once virtual machine allocation is requested, the cumulative mean resource usage in a physical server is calculated, thereby identifying the physical server targeted for allocation. Specifically, information on the physical server whose resource utilization is the minimum for each resource is ascertained.

The resource utilization data of users are determined by computing the cumulative mean usage after identifying the resource utilization of users for each individual resource. This identification is based on the resource utilization log data of a virtual machine and a user.

The prediction data of resource utilization for each individual resource in a virtual machine operated in a physical server are ascertained using the log data of resource utilization in virtual machines. To avoid the duplicate maximum utilization of specific resources, the prediction data of resource utilization in a virtual machine and the prediction data of resource utilization by a user who requests allocation are compared. The resource utilization prediction data of a physical server after a virtual machine is allocated to a user are derived. Figure 4 illustrates the pseudocode executed by the virtual machine placement algorithm under pooled allocation.



```
Algorithm 2 Virtual Machine Allocation Algorithm
AllocationVM(PMList, VMList, $USER_n$)
    Compute resource utilization of $USER_n$
    for each i ∈ PMList
        Compute resource utilization of $PM_i$
        PMUList.add($PM_i U_N$)
    end for
    Sort_ASC(PMUList)
    for each i ∈ PMUList
        for each n ∈ $PM_i$
            Compute resource utilization of $VM_n$ running on $PM_i$
            VMUList.add($VM_n U_N$)
        end for
        Sort_DESC(VMUList)
        if Compare $User_n U_N$ with $VM_n U_N$
            in the VMUList == Most_Resource_Utilization then
            continue
        end if
    end for
    Allocate $VM_j$ running on $PM_k$ to $USER_n$
    Compute resource utilization of $PM_k$
```

Figure 4. Pseudocode of the virtual machine allocation algorithm

IV. Experimental results and analyst

The performance and execution results of the algorithm in terms of virtual machine allocation and placement were evaluated on the basis of the following metrics: load variance in resource utilization in a physical server, standard deviation (SD) of utilization in a physical server, and the number of overload hosts. Experiments were initiated with the collection of the resource utilization data of physical servers and virtual machines in a DaaS environment for companies, after which the algorithms were applied, and load balancing simulations based on the collected data were run on CloudSim [8].

Tables 2 and 3 present the configuration environment of physical servers and the resource quota of virtual machines running in each physical server. As indicated in Table 2, the configuration environment for 10 physical servers was divided and presented by Physical Machine Identification (PMID). The CPU cores in the PMID 1-5 and PMID 6-10 physical servers amounted to 20 and 10, respectively. The memory capacities of these servers were 320 and 224 GB, respectively, pointing to a heterogeneous environment with different hardware specifications. Table 3 displays information on resource utilization by virtual machines in physical servers. The vCPU cores of all virtual machines amounted to two. A memory of 4 GB was statically assigned to the virtual machines in PMID 1-5, whereas a memory of 1 to 3 GB was dynamically allocated to the virtual machines operating in PMID 6-10.

TABLE II. CONFIGURATION ENVIRONMENT OF PHYSICAL SERVER

Item	Physical Server Type	
	Type 1	*Type 2*
PMID	1,2,3,4,5	6,7,8,9,10
Model	Lenovo HR630X	Fujitsu RX2540 M1
CPU	Intel Gold 6132 2.6 GHz*2EA	Intel Xeon E5-2660 v3 2.60 GHz*2EA
CPU cores	20*2EA	10*2EA
Memory	320GB	224GB

TABLE III. RESOURCE QUOTA OF VIRTUAL MACHINE IN PHYSICAL SERVER

Item	Virtual Machine Type	
	Type 1	*Type 2*
PMID	1,2,3,4,5	6,7,8,9,10
vCPU	2 Core	2 Core
Memory	4 GB (static)	1~3 GB (dynamic)
Disk	30GB	30GB

The experiments involved executing the algorithm on the basis of the collected log data of resource utilization in each virtual machine over the DaaS environment for companies. Table 4 summarizes the number of virtual machines running in each physical server and the CPU resource utilization in each of these servers.

TABLE IV. NUMBER OF VIRTUAL MACHINES IN ALL PHYSICAL SERVERS AND CPU RESOURCE UTILIZATION

PMID	Number of VM	CPU Utilization of Virtual Machines			
		mean	Maximum	Minimum	SD
1	73	31.51	80.51	16.54	15.84
2	71	29.29	60.51	4.97	13.96
3	69	26.72	55.05	13.28	12.13
4	74	30.90	61.77	16.87	14.26
5	72	30.64	76.36	14.82	14.94

PMID	Number of VM	CPU Utilization of Virtual Machines			
		mean	Maximum	Minimum	SD
6	73	32.59	60	10	9.34
7	66	19.95	50	6	7.95
8	65	15.30	63	3	8.14
9	63	17.15	47	4	6.70
10	66	14.45	55	3	8.03

Table 5, Figure 5 and Figure 6 shows the results on changes in CPU utilization in all physical servers after the proposed algorithm was deployed. The least connection (LC) algorithm was used to balance load in the existing operating environment. The value of the entire CPU utilization data of a virtual machine and a user did not change; thus, the mean CPU utilization of the physical server on the basis of the CPU utilization of the virtual machine was 24.85% both before and after the activation of the algorithm. Although the maximum CPU utilization of the physical server that exhibited this usage was 80.51% before the execution of the algorithm, the mean load of the physical server was reduced by changing the allocation of the virtual machine with the largest resource utilization. This modification caused overload in the physical server and spilled over to the user who employed resources to a minimal extent by running the algorithm. As a result, the CPU utilization of the physical server that showed the maximum CPU utilization decreased to 62.48%, which is a 18.03% reduction compared with the level before the application of the algorithm. These outcomes reduced the maximum resource utilization of the physical server, thereby increasing resource availability. Ultimately, these findings reflect that performance degradation and service outage because of overload in physical servers can be prevented.

TABLE V. CPU UTILIZATION OF ALL PHYSICAL SERVERS AFTER ALGORITHM

Item	Algorithm	
	Least Connection	***Ours***
Mean CPU utiliazation	24.85	24.85
Maximum CPU utilization	80.51	62.48
Minimum CPU utilization	3	10.31
Mean SD	5.11	2.95
Maximum mean SD	14.54	6.89
Minimum mean SD	1.93	1.71

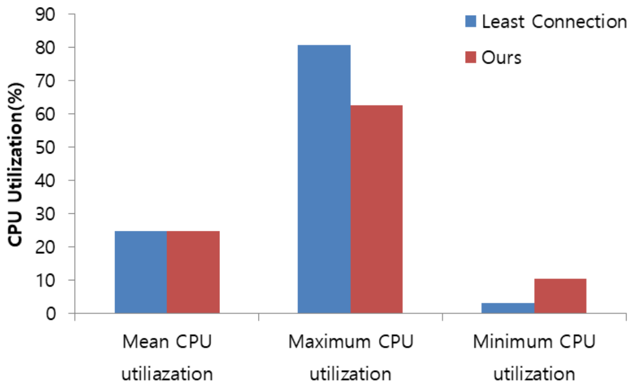


Figure 5. Changes in CPU utilization in all physical servers (maximum and minimum utilizations)

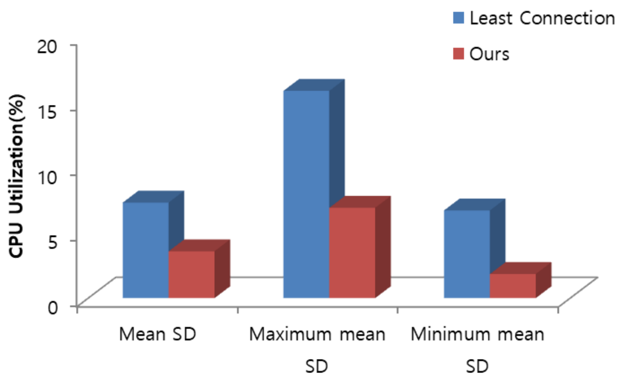


Figure 6. Changes in CPU mean standard deviation in all physical servers

The algorithm execution results showed that the mean standard deviation of all physical servers decreased from 5.11% to 2.95% and that the maximum mean CPU utilization declined from 14.54% to 6.89%. These results indicate that the standard deviation of CPU utilization between physical servers per hour significantly decreased overall compared with the level prior to algorithm execution. This resulted in good load balancing in the physical servers. The algorithm prevented an increase in load due to the arbitrary and exponential rise in the resource utilization of physical servers. This was accomplished through the elimination of duplicate virtual machines, whose utilization of the specific resources of virtual machines in a particular server was at a maximum. Given that the maximum resource utilization was stably maintained after load balancing execution through virtual machine allocation and placement, the migration of virtual machines was unnecessary, and the mapping relationship between physical servers and virtual machines could be reliably maintained. Figure 7 depicts the change in the standard deviation of mean CPU utilization in all physical servers after algorithm deployment.

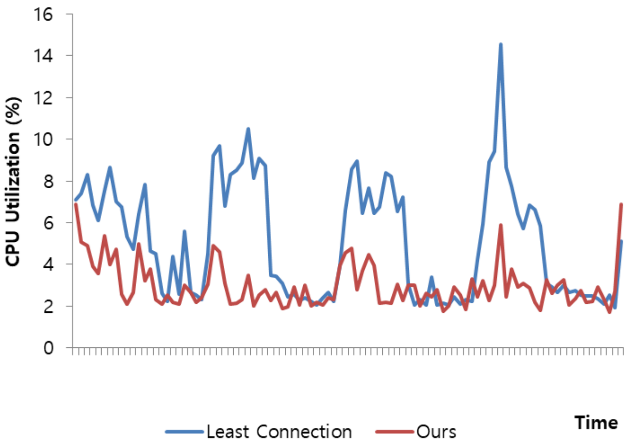


Figure 7. Changes in CPU mean standard deviation of CPU utilization in all physical servers

V. Conclusion

This study analyzed the usage patterns of users on the basis of the resource utilization log data of physical servers and virtual machines as DaaS is provided to companies via virtualization technology. This analysis enabled the deterministic prediction of resource utilization in physical servers using resource demands from virtual machines and users. The research was aimed at developing a technique for providing highly efficient DaaS, for which the loads borne by physical servers is balanced through resource management via a virtual machine management technique. This technique ascertains virtual machine allocation and placement on the grounds of an empirical knowledge-based strategy for predicting resource utilization in physical servers.

The study also put forward a virtual machine placement algorithm, by which virtual machines are placed through the selection of physical servers with the least resource utilization. The selection commences with a comparison of the cumulative mean resource utilization levels of all physical servers during virtual machine placement. The algorithm allocates virtual machines by calculating the resource utilization of physical servers on the basis of the cumulative mean resource utilization of users when virtual machines are placed in physical servers. The resource utilization data of physical servers, virtual machines, and users in a DaaS environment were collected to conduct experiments on algorithm performance.

Future plans for this work include applying modules to actual environments and implementing the proposed virtual machine management in real-world service environments to determine whether improvements to the technique are necessary, as reflected by experimental results.

References

[1] ETRI, “Trends of Virtual Desktop Infrastructure Technology,” Electronics and Telecommunications Trends, 2013.

[2] ETRI, http://www.etri.re.kr

[3] T. Wood, P. Shenoy, A. Venkataramani and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," Computer Networks Journal(ComNet), vol. 53, no. 17, 2009.

[4] B. Li and Y. Wang, "An Distributed Virtual Machine Placement Algorithm for Balanced Resource Utilization and Low Energy Consumption," in MATEC Web of Conferences, 2018.

[5] Y. Zhao and W. Huang, "Adaptive Distributed Load Balancing Algorithm based on Live Migration of Virtual Machines in Cloud," in International Joint Conference on INC, IMS and IDC, 2009.

[6] M. Xu, W. Tian and R. Buyya, "A survey on load balancing algorithms for virtual machines placement in cloud computing," arXiv:1607.06269v3, 2017.

[7] Tilon, http://www.tilon.com

[8] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, vol. 41, no. 1, pp. 23-50, 2011.