J Inf Process Syst, Vol.14, No.6, pp.1480~1493, December 2018
https://doi.org/10.3745/JIPS.02.0102

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

An Intelligent Residual Resource Monitoring Scheme in Cloud Computing Environments

JongBeom Lim*, HeonChang Yu**, and Joon-Min Gil***

Abstract

Recently, computational intelligence has received a lot of attention from researchers due to its potential applications to artificial intelligence. In computer science, computational intelligence refers to a machine's ability to learn how to compete various tasks, such as making observations or carrying out experiments. We adopted a computational intelligence solution to monitoring residual resources in cloud computing environments. The proposed residual resource monitoring scheme periodically monitors the cloud-based host machines, so that the post migration performance of a virtual machine is as consistent with the pre-migration performance as possible. To this end, we use a novel similarity measure to find the best target host to migrate a virtual machine to. The design of the proposed residual resource monitoring scheme helps maintain the quality of service and service level agreement during the migration. We carried out a number of experimental evaluations to demonstrate the effectiveness of the proposed residual resource monitoring scheme. Our results show that the proposed scheme intelligently measures the similarities between virtual machines in cloud computing environments without causing performance degradation, whilst preserving the quality of service and service level agreement.

Keywords

Cloud Computing, Clustering, Computational Intelligence, Resource Monitoring

1. Introduction

Computational intelligence and artificial intelligence techniques represent a new paradigm for managing computer servers in cloud computing environments [1-4]. Without computational intelligence and artificial intelligence techniques, human beings are required to monitor the status of computational resources—e.g., CPU utilization, memory usage, input/output (I/O) performance, etc. This human intervention is unavoidable because system failures can occur at any time and human beings are required to take measures in the case of such failures [5,6]. Moreover, intensive human intervention is almost inevitable for making the meaningful observations that are necessary to effectively manage large-scale servers in cloud computing systems [7,8]. The recent development of computational intelligence techniques shed some light on how to deal with these problems effectively [9-11].

Efficient management of cloud computing resources includes the management of connected services

Manuscript received August 31, 2018; first revision October 2, 2018; accepted October 22, 2018.
Corresponding Author: Joon-Min Gil (jmgil@cu.ac.kr)
* Dept. of Game & Multimedia Engineering, Korea Polytechnic University, Siheung, Korea (jblim@kpu.ac.kr)
** Dept. of Computer Science & Engineering, Korea University, Seoul, Korea (yuhc@korea.ac.kr)
*** School of Information Technology Engineering, Daegu Catholic University, Gyeongsan, Korea (jmgil@cu.ac.kr)

for Internet of Things (IoT) devices and big data analytics. IoT devices, which have limited CPUs and batteries, can offload their processing to cloud computing resources [12,13]. For big data analytics, unlimited cloud computing resources offer fast, efficient, low-cost processing services on demand [14]. These benefits can be achieved by efficient resource management of cloud computing environments.

In this paper, we propose an intelligent residual resource monitoring scheme for cloud computing environments. The proposed residual resource monitoring scheme periodically monitors host machines in cloud computing environments, so that the post migration performance of a virtual machine is as similar to the pre-migration performance as possible. To this end, we use a novel similarity measure to find the best target host to migrate a given virtual machine to. The design of the proposed residual resource monitoring scheme helps maintain quality of service (QoS) and service level agreement (SLA) during the migration process.

Our scheme differs from the traditional monitoring schemes described in the literature because we have designed and implemented a novel scheme for monitoring the residual resources of cloud-based host machines. This ensures that the performance of the pre-migration virtual machine is as close as possible to that of the post-migration virtual machine. By leveraging the proposed monitoring scheme, dynamic cloud computing consolidation can be undertaken without human intervention, while preserving the QoS and SLA.

The major contributions of this paper are summarized as follows:

(i) We designed an intelligent residual resource monitoring scheme based on the current status of the host machines in cloud computing environments.

(ii) We developed a novel metric to calculate the similarity of the residual resources of virtual machines for migration and cloud consolidation.

(iii) We formulated a migration problem that takes into account the QoS and SLA, and implemented a dynamic residual resource monitoring algorithm by considering the scalability.

(iv) We undertook a comprehensive analysis and performance evaluation to show the effectiveness of the proposed monitoring scheme in a real-world scenario.

The remainder of the paper is organized as follows: After reviewing related work in Section 2, we describe the system model and formally define the problem addressed in Section 3. The proposed intelligent residual resource monitoring scheme for cloud computing environments is presented in Section 4. In Section 5, we present the results of our performance evaluation, with comparisons to previous systems. Finally, in Section 6, we conclude the paper with future research directions.

2. Related Work

The monitoring of computing resources is one of the fundamental problems of distributed and cloud computing environments. Because cloud computing environments generally have no shared memory, these systems are not trivial to monitor. We can detect malware, intrusion, and other types of misbehavior by monitoring computing resources [15,16]. Another reason for monitoring cloud computing environments is to migrate virtual machines in a secure way [17]. More specifically, when a virtual machine is scheduled for migration to an overloaded host machine, the QoS and SLA may not be guaranteed.

There are some centralized monitoring solutions for cloud computing environments [18-20]. The

disadvantage of centralized monitoring schemes is the lack of scalability. In other words, when the number of nodes in the system increases, the overhead also increases. This may result in bottlenecks at single points of failure.

The authors of [21] proposed a distributed monitoring architecture for cloud computing systems. However, they did not detail how to map virtual machines and host machines in dynamic environments. Moreover, they require software agents, which are not suitable for our system design. VMDriver [22] is a fine-grained monitoring system for virtualized environments. The design is based on dividing event interception between the virtual machine monitor layer and the virtual machine semantic reconstructions, with implementation of a monitoring driver in Xen [23]. This driver uses a virtual machine monitor to gather the required information.

In [24], the authors focused on analyzing various monitoring architectures to determine their pros and cons. Therefore, cloud computing administrators can make choices based on the requirements imposed by their architectures. Aceto et al. [10] surveyed monitoring architectures for commercial solutions in detail. However, the internal implementations of these architectures are invisible.

The authors of [25] presented a monitoring framework based on peer to peer architecture, similar to our architectural design. However, the design goal differs in that our purpose for monitoring cloud resources is to ensure that the post-migration performance of a virtual machine is as similar as possible to the pre-migration performance.

Recently, Wang et al. [26] proposed a monitoring system with self-adaptive properties for cloud computing environments. The parameters of this system are adjusted dynamically based on a reliability metric. However, the aim of this monitoring technique is to detect anomalies.

In this regard, none of the discussed monitoring architectures fulfil our requirements. The basic requirement of our intelligent residual resource monitoring scheme is that it must be easily integrated into cloud computing infrastructures, without human intervention, so that the QoS and SLA are maintained during the migration. In the next section, we describe the system model and explain how our design goal can be achieved intelligently in cloud computing environments without human intervention.

3. System Model and Problem Definition

An informal definition of computational intelligence is an approach based on techniques inspired by nature to solve various problems, to which traditional techniques cannot be applied [27-29]. To monitor the residual resources of host machines in an intelligent way, we consider the hosts as a set of n nodes, where each node is connected by logical communication channels. Because each host machine has distinct resource properties (CPU utilization, memory usage, I/O performance, etc.), the performance of a virtual machine may differ after it is migrated to another host machine.

Fig. 1 shows the performance degradation problem induced by virtual machine migration. The relative performance of host A is 1, while that of host B is 0.6. Before migration, the virtual machine resides on host A. If the virtual machine is scheduled for migration to host B, the performance will not be as good as on host A. In fact, the relative performance of the virtual machine is 0.6. In this case, it is difficult for the virtual machine to guarantee the QoS and SLA on host B.

On the other hand, the opposite case can also be a problem. More specifically, when a virtual machine

running on host B, on which there is no problem regarding the QoS and SLA, is scheduled for migration to host A, another virtual machine that is required for the QoS and SLA may not be migrated to host A because the number of virtual machines running on the host machine is limited. The goal of the proposed residual resource monitoring scheme for cloud computing environments is to identify a similar host machine to migrate the virtual machine on to. This will maintain the pre-migration performance of the virtual machine as much as possible.

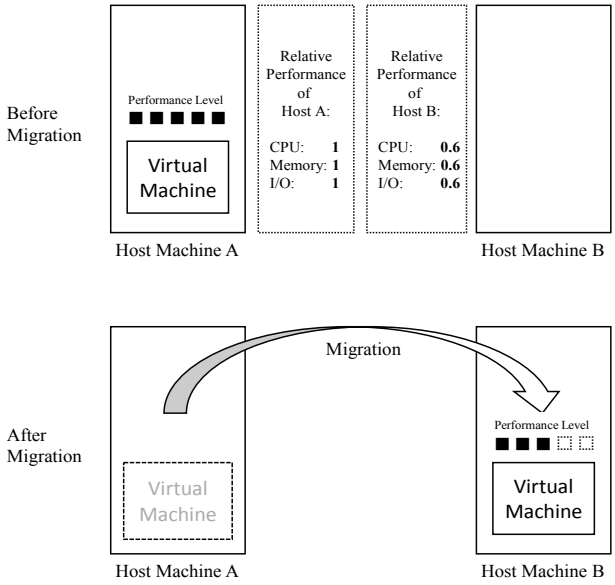

Fig. 1. Performance degradation problem after virtual machine migration.

4. Proposed Methodology and Algorithms

In this section, we describe the proposed residual resource monitoring scheme in cloud computing environments. Two things should be considered for cloud consolidation to effectively migrate a virtual machine. The first is to monitor the unused resources on the potential host (the number of vCPUs, memory, storage, etc.). The migration cannot be allowed if the unused resources do not meet the virtual machine's minimal requirements. The other requirement is to measure the performance of the host machines in terms of their unused resources. This is important for maintaining the QoS and SLA, as described in the previous section.

Fig. 2 shows the monitoring architecture for cloud computing environments. The physical resources (CPU, memory, storage, etc.) are virtualized by the hypervisor or virtual machine monitor. In the hypervisor level, the virtualized resources can be managed for efficient server consolidation (e.g., real-time virtual machine scheduler, QoS monitoring module for services, energy-usage prediction module, service type classification, etc.). At a higher level, the cloud platform performs the virtual machine provisioning, efficient and energy-aware virtual machine placement, and server consolidation. The main advantage of our monitoring architecture is that the management of the cloud computing systems can be performed in an intelligent way, without human intervention.

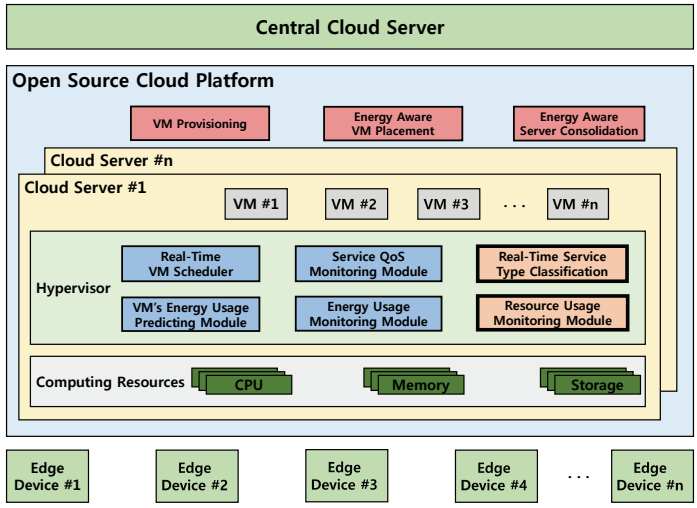

Fig. 2. The monitoring architecture in cloud computing environments.

When measuring the performance of the host machines' unused resources, we consider the following two metrics: CPU and I/O performance. We predict that the memory performance will not significantly affect the performance of the virtual machines. Meanwhile, the CPU and I/O configurations may affect the performance of virtual machines because of variation between models—e.g., cache size for CPUs, and hard disk drive (HDD) or solid state drive (SDD) for I/O.

We use the resource cosine similarity between two nodes defined in Eq. (1) to measure the similarities between the host machines' unused resources:

$$ResCos(node_1, node_2) = \frac{|Res_{node1}| \cap |Res_{node2}|}{\sqrt{|Res_{node1}| \times |Res_{node2}|}} \tag{1}$$

The *ResCos* score between two nodes increases when their resource properties are similar. Our preliminary experiments have shown that *ResCos* outperforms a simple measure that counts the number of common resource properties.

Furthermore, to quantify the similarities, we calculate $|ResCos(node1, node2)|$, as defined in Eq. (2).

$$\left| \frac{|Res_{node1}| \cap |Res_{node2}|}{\sqrt{|Res_{node1}| \times |Res_{node2}|}} \right| \tag{2}$$

By calculating the metrics defined in Eqs. (1) and (2), the proposed monitoring scheme is able to identify a suitable host machine for migration. In addition, we employ the tuning function to adjust the importance of the properties. This is defined by Eq. (3). The tuning function allows the monitoring scheme to prioritize between Eqs. (1) and (2).

$$\begin{aligned} &TunedScore(node_1, node_2) = \\ &(\alpha \times |ResCos(node_1, node_2)|) + \\ &(\beta \times ResCos(node_1, node_2)) \end{aligned} \tag{3}$$

To calculate the value of *ResCos* for n nodes, the monitor manager must perform $\frac{n(n-1)}{2}$ operations, so the complexity is $O(n^2)$. We can reduce the complexity by letting each node evaluate the scoring functions. Each node informs the manager when it has finished evaluating its scoring functions. The monitor manager then uses the collected information as required to migrate a virtual machine. This offloading feature enables the cloud computing system to scale the number of nodes.

Algorithm 1. The proposed monitoring algorithm.

Input: VM_i, PM_j, where $\forall i \in \{1, 2, \ldots, n\}$ and $\forall j \in \{1, 2, \ldots, m\}$
Output: Optimized map $(VM_i, PM_j)_{best}$
begin // It is periodically performed and VM migration is scheduled.
  for each VM_i, where $\forall i \in \{1, 2, \ldots, n\}$
    Call VMMonitor(VM_i); // We let each VM perform the function.
  end for
  for each PM_j, where $\forall j \in \{1, 2, \ldots, m\}$
    Call MonitorResidualResource(PM_j); // We let each PM perform the function.
  end for
end
begin when a VM (VM_i) is scheduled for migration
  $ResVM_i \leftarrow$ Retrieve monitored information of VM_i.
  $bestPM \leftarrow null$;
  for each PM_j, where $\forall j \in \{1, 2, \ldots, m\}$
    Calculate $ResCos_j(ResVM_i, ResPM_j)$;
    Calculate $|ResCos_j(ResVM_i, ResPM_j)|$;
    $TunedScore_j \leftarrow TunedScore(ResVM_i, ResPM_j)$;
    if $TunedScore_j > bestPM$ **then**
      $bestPM \leftarrow PM_j$;
    end if
  end for
  return map(VM_i, $bestPM$);
end
function VMMonitor(VM_i);
  $ResVM_i.SLA \leftarrow VM_i.requirement$; // It stores the resource and SLA information.
  $ResVM_i.AllocatedVCPU \leftarrow VM_i.VCPU$; // It stores vCPU's type and the number of vCPUs.
  $ResVM_i.AllocatedMemory \leftarrow VM_i.Memory$; // It stores allocated memory.
  $ResVM_i.AllocatedStorage \leftarrow VM_i.Storage$; // It stores storage's type and size.
end function
function MonitorResidualResource(PM_i);
  $ResPM_j.ResidualCPU \leftarrow PM_j.CPU$; // It stores CPU type and available CPU information.

Algorithm 1 shows the pseudocode of the proposed monitoring algorithm, which is based on the similarity measure. The properties of the virtual machine (VM_i) and physical machine (PM_j) are taken as input. Note that in the algorithm, we use subscript *i* for virtual machines and *j* for physical machines. The output is the mapping information between the virtual machines and the physical machines.

The proposed monitoring scheme is designed to periodically perform the monitoring procedure (lines 1–8). Unlike the previous monitoring techniques, we offload the monitoring process to each node (both virtual and physical machine); therefore, the proposed monitoring scheme is scalable in terms of the number of cloud computing nodes. The procedures followed by the *VMMonitor*() and *MonitorResidualResource*() functions are detailed on lines 22–27 and 28–32, respectively.

The procedure implemented on lines 9–21 is performed when a virtual machine is scheduled for migration. The input to this procedure is the virtual machine information and the output is the mapping information between the virtual machine and the physical machines. This enables the cloud platform to migrate the virtual machine to the physical machine.

After retrieving the virtual machine information, it calculates the similarity measure to find the best physical machine for migration. At each iteration (lines 12–19), our algorithm stores the best physical machine according to our similarity measure. Next, it returns the mapping information between the virtual machine and the physical machine.

Since our monitoring algorithm evaluates the residual resources of the physical machines that are closest to those required by the virtual machines, our procedure has the following advantages: First, the post-migration performance of the virtual machine is as similar as possible to the pre-migration performance. Hence, the QoS and SLA will be guaranteed. Second, additional energy savings can be realized through the cloud consolidation enabled by our monitoring scheme.

For instance, suppose that there is a virtual machine on a physical machine (*PM_A*) and that the virtual machine's score is 0.3. In this case, our proposed monitoring scheme finds a physical machine (*PM_B*) that can barely afford the virtual machine (i.e., the residual score is close to 0.3). Therefore, *PM_A* will be turned off to save energy, while *PM_B* runs more virtual machines without performance degradation.

Our monitoring scheme can be applied to various implementation patterns. For example, the cloud coordinator commands and collects monitoring information on each of the nodes in the system. This monitoring information can be used when selecting a target machine for live migration. Since the complexity of the algorithm increases linearly according to the number of machines, our monitoring scheme can be applied in a scalable system with low overheads.

5. Performance Evaluation

In this section, we present experimental results that demonstrate the performance of the proposed monitoring scheme for maintaining the QoS and SLA. Table 1 shows the experimental settings used for evaluating the performance of the monitoring scheme. These types of cloud tasks are computationally intensive tasks used for multimedia and big data processing. Although we employed an equal number of physical and virtual machines for the proof of concept, different scenarios can also be used.

Table 1. Experimental settings

Parameter	Value
Number of physical machines	1,000
Number of virtual machines	1,000
Host memory (MB)	8192
Host bandwidth (GB/s)	1
The number of CPUs	[2, 4]
The number of vCPU	[1, 4]
vCPU capacity (MIPS)	[500, 2000]
vRAM (MB)	[256, 2048]
Bandwidth (MB/s)	[100, 1000]
The number of cloud tasks	[100, 500]
The required MIPS for tasks	[100, 20000]
File size of tasks (MB)	[100, 500]
Output file size (MB)	[40, 80]

To verify the scalability of the proposed algorithm, we set the number of physical and virtual machines to 1,000. In other words, there is one virtual machine running on each physical machine in the cloud computing system. We configured hardware settings by taking into account the relative performance of the physical machines and the virtual machines (e.g., memory, bandwidth, the number of CPUs, etc.).

Fig. 3 shows the relative performance indices of the physical machines. Note that the higher the value of the relative performance index, the better. These values are in the range between 0 and 1. This setting helps us to verify that the proposed monitoring scheme identifies a suitable target machine for migration.

Fig. 4 shows the difference in the relative performance index after finding the target migration machine. Note that the results are depicted on a logarithmic scale and confirm that our intelligent residual resource monitoring scheme works well without incurring SLA violations. The average difference in the relative performance index after identifying a suitable target migration machine is 0.000492899.

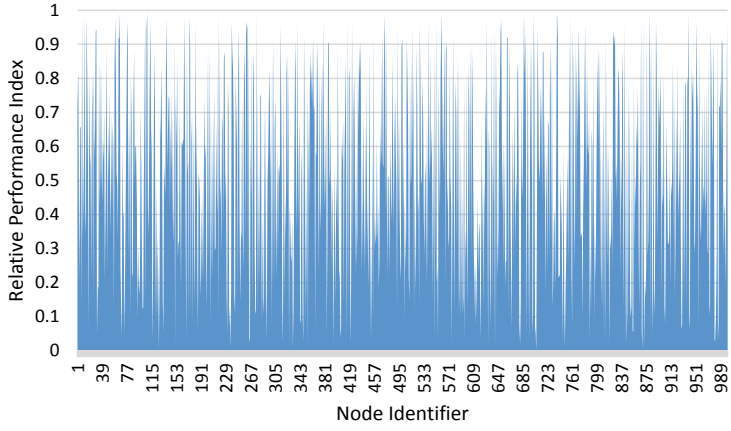

Fig. 3. Relative performance indices of physical machines.

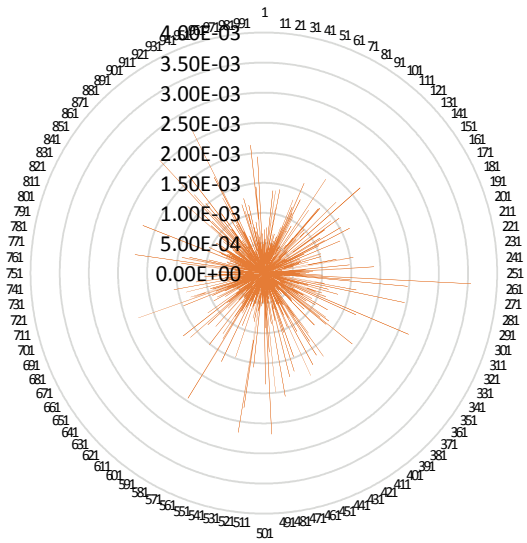

Fig. 4. Difference in the relative performance indices before versus after finding the target migration machine with the proposed scheme.

Fig. 5 shows the count used to select a target machine from the set of all nodes. The results were between 0 and 2 and the counts were 256, 488, and 256 for 0, 1, and 2, respectively. When the count value was greater than or equal to 2, the migration process was managed by the cloud platform and the monitoring information relating to the target physical machine should have been updated. About half of the machines were selected as migration targets from the nodes at least once.

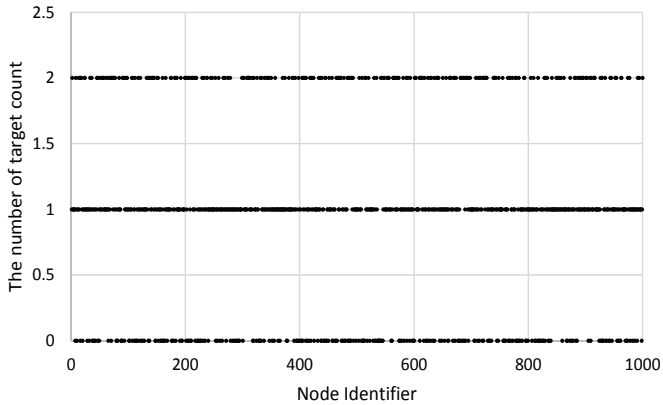

Fig. 5. The count used to select the target machine from the available nodes.

We compared migrations with and without implementing our proposed monitoring scheme by measuring the effect of not using the proposed scheme on the relative performance index, as shown in Fig. 6. The maximum and minimum values obtained are 0.988093748 and -0.984641698, respectively. This implies that the performance is hugely different before versus after migration. This will result in SLA violations or increased energy consumption.

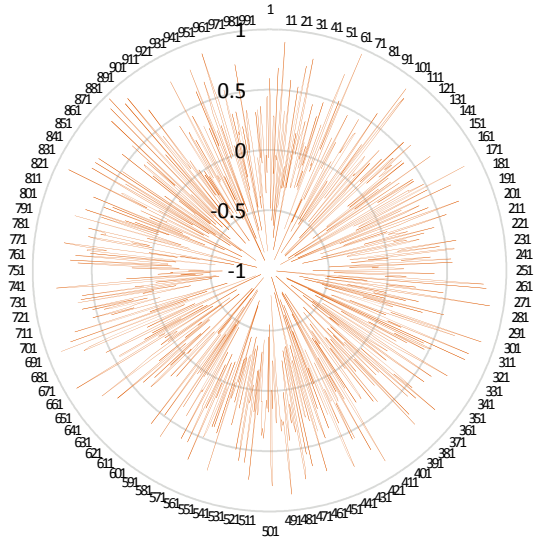

Fig. 6. Difference in the relative performance indices before versus after finding the target migration machine without the proposed scheme.

Fig. 7 shows the difference in the relative performance indices obtained with versus without the proposed scheme. The average of the difference is 0.337265836 and the standard deviation is 0.240753321. The results of this comparison validate the effectiveness of our intelligent residual resource monitoring scheme.

We investigated how SLA violations occur when the proposed scheme is not used by measuring the number of virtual machines that cause violations, as shown in Fig. 8. We categorized the SLA violations from level 0 to level 5. Level 5 indicates that the performance decreases by more than 50% (level 4 for 40%, and so on). Level 0 includes all post-migration performance degradation. The number of SLA violations without the proposed scheme for levels 5, 4, 3, 2, 1, and 0 was 131, 187, 253, 318, 390, and 485, respectively. The number of level 1 SLA violations was 0 when the proposed scheme was used.

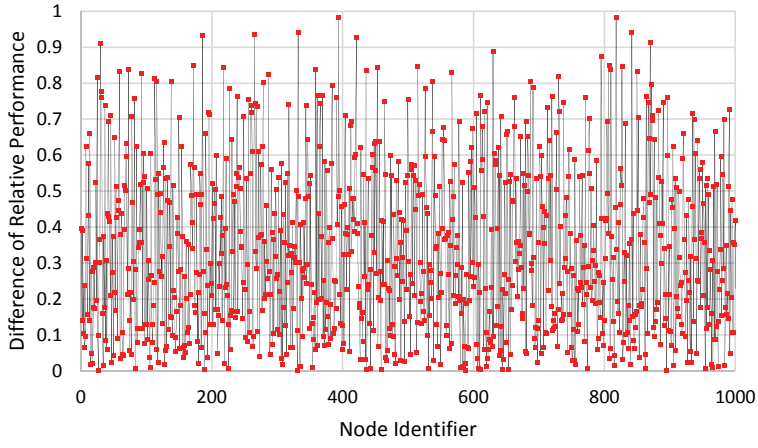

Fig. 7. Difference in the relative performance indices with versus without the proposed scheme.

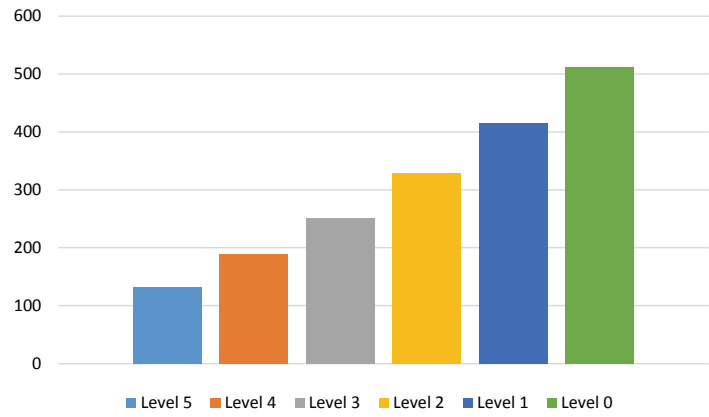

Fig. 8. Service level agreement violation without the proposed scheme.

6. Conclusion

In this paper, we proposed an intelligent residual resource monitoring scheme for cloud computing environments. The proposed monitoring scheme can effectively find a host machine for migration, so that the post migration performance of a virtual machine is consistent with its pre-migration performance. By collecting monitoring information from host machines in cloud computing environments, the QoS and SLA can be maintained without violation. We confirmed that the proposed scheme reduces the incidence of post-migration performance degradation by more than 90%. Thus, our monitoring scheme is suitable for SLA sensitive workloads, including real-time and multimedia applications. In future work, we will conduct an extensive performance evaluation for various scalability settings and investigate the use of artificial intelligence techniques to respond to failures in dynamic environments.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2018R1D1A1B07045838 and NRF-2016R1D1A3B03933370). The author to whom correspondence should be addressed is Joon-Min Gil.

References

[1] C. Jatoth, G. R. Gangadharan, and R. Buyya, "Computational intelligence based QoS-aware web service composition: a systematic literature review," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 475-492, 2017.

[2] N. K. Gondhi and A. Gupta, "Survey on machine learning based scheduling in cloud computing," in *Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*, Hong Kong, China, 2017, pp. 57-61.

[3] D. Gabi, A. S. Ismail, A. Zainal, and Z. Zakaria, "Quality of service (QoS) task scheduling algorithm with Taguchi orthogonal approach for cloud computing environment," in *Recent Trends in Information and Communication Technology*. Cham: Springer, 2018, pp. 641-649.

[4] Z. H. Zhan, X. F. Liu, Y. J. Gong, J. Zhang, H. S. H. Chung, and Y. Li, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Computing Surveys*, vol. 47, no. 4, article no. 63, 2015.

[5] P. Di Sanzo, A. Pellegrini, and D. R. Avresky, "Machine learning for achieving self-* properties and seamless execution of applications in the cloud," in *Proceedings of 2015 IEEE 4th Symposium on Network Cloud Computing and Applications (NCCA)*, Munich, Germany, 2015, pp. 51-58.

[6] A. N. Toosi, R. N. Calheiros, and R. Buyya, "Interconnected cloud computing environments: challenges, taxonomy, and survey," *ACM Computing Surveys*, vol. 47, no. 1, article no. 7, 2014.

[7] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "big data" on cloud computing: review and open research issues," *Information Systems*, vol. 47, pp. 98-115, 2015.

[8] M. Silva, M. R. Hines, D. Gallo, Q. Liu, K. D. Ryu, and D. Da Silva, "CloudBench: experiment automation for cloud environments," in *Proceedings of 2013 IEEE International Conference on Cloud Engineering (IC2E)*, Redwood City, CA, 2013, pp. 302-311.

[9] N. Bessis, E. Asimakopoulou, T. French, P. Norrington, and F. Xhafa, "The big picture, from grids and clouds to crowds: a data collective computational intelligence case proposal for managing disasters," in *Proceedings of 2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, Fukuoka, Japan, 2010, pp. 351-356.

[10] G. Aceto, A. Botta, W. de Donato, and A. Pescape, "Cloud monitoring: a survey," *Computer Networks*, vol. 57, no. 9, pp. 2093-2115, 2013.

[11] M. T. Khorshed, A. B. M. S. Ali, and S. A. Wasimi, "Monitoring insiders activities in cloud computing using rule based learning," in *Proceedings of 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, Changsha, China, 2011, pp. 757-764.

[12] P. K. Sharma, J. H. Ryu, K. Y. Park, J. H. Park, and J. H. Park, "Li-Fi based on security cloud framework for future IT environment," *Human-centric Computing and Information Sciences*, vol. 8, article no. 23, 2018.

[13] W. Song, N. Feng, Y. Tian, S. Fong, and K. Cho, "a deep belief network for electricity utilisation feature analysis of air conditioners using a smart IoT platform," *Journal of Information Processing Systems*, vol. 14, no. 1, pp. 162-175, 2018.

[14] M. R. Mesbahi, A. M. Rahmani, and M. Hosseinzadeh, "Reliability and high availability in cloud computing environments: a reference roadmap," *Human-centric Computing and Information Sciences*, vol. 8, article no. 20, 2018.

[15] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Human-centric Computing and Information Sciences*, vol. 8, article no. 3, 2018.

[16] R. Zhang and X. Xiao, "Study of danger-theory-based intrusion detection technology in virtual machines of cloud computing environment," *Journal of Information Processing Systems*, vol. 14, no. 1, pp. 239-251, 2018.

[17] S. B. Rathod and V. K. Reddy, "NDynamic framework for secure VM migration over cloud computing," *Journal of Information Processing Systems*, vol. 13, no. 3, pp. 476-490, 2018.

[18] J. Shao, H. Wei, Q. Wang, and H. Mei, "A runtime model based monitoring approach for cloud," in *Proceedings of 2010 IEEE 3rd International Conference on Cloud Computing*, Miami, FL, 2010, pp. 313-320.

[19] H. Huang and L. Wang, "P&P: a combined push-pull model for resource monitoring in cloud computing environment," in *Proceedings of 2010 IEEE 3rd International Conference on Cloud Computing*, Miami, FL, 2010, pp. 260-267.

[20] M. Rak, S. Venticinque, T. Mahr, G. Echevarria, and G. Esnal, "Cloud application monitoring: the mOSAIC approach," in *Proceedings of 2011 IEEE 3rd International Conference on Cloud Computing Technology and Science*, Athens, Greece, 2011, pp. 758-763.

[21] M. Andreolini, M. Colajanni, and M. Pietri, "A scalable architecture for real-time monitoring of large information systems," in *Proceedings of 2012 Second Symposium on Network Cloud Computing and Applications*, London, UK, 2012, pp. 143-150.

[22] G. Xiang, H. Jin, D. Zou, X. Zhang, S. Wen, and F. Zhao, "VMDriver: a driver-based monitoring mechanism for virtualization," in *Proceedings of 2010 29th IEEE Symposium on Reliable Distributed Systems*, New Delhi, India, 2010, pp. 72-81.

[23] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, & A. Warfield, "Xen and the art of virtualization," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, Bolton Landing, NY, 2003, pp. 164-177.

[24] Y. Sandoval, G. Gallizo, and M. Curiel, "Evaluation of monitoring tools for cloud computing environments," in *Proceedings of 2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI)*, Medellin, Colombia, 2012, pp. 1-10.

[25] B. Konig, J. M. Calero, and J. Kirschnick, "Elastic monitoring framework for cloud infrastructures," *IET Communications*, vol. 6, no. 10, pp. 1306-1315, 2012.

[26] T. Wang, J. Xu, W. Zhang, Z. Gu, and H. Zhong, "Self-adaptive cloud monitoring with online anomaly detection," *Future Generation Computer Systems*, vol. 80, pp. 89-101, 2018.

[27] G. N. Yannakakis and J. Togelius, "A panorama of artificial and computational intelligence in games," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 7, no. 4, pp. 317-335, 2015.

[28] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, "Transfer learning using computational intelligence: a survey," *Knowledge-Based Systems*, vol. 80, pp. 14-23, 2015.

[29] J. A. Khan, M. A. Z. Raja, M. M. Rashidi, M. I. Syam, and A. M. Wazwaz, "Nature-inspired computing approach for solving non-linear singular Emden–Fowler problem arising in electromagnetic theory," *Connection Science*, vol. 27, no. 4, pp. 377-396, 2015.

JongBeom Lim https://orcid.org/0000-0001-8954-2903

He received B.S. degree in information and communication from Baekseok University, Korea in 2009. In 2011 and 2014, he received M.S. and Ph.D. degrees in computer science and education from Korea University, Korea, respectively. From 2015 to 2017, he was a visiting professor with the IT Convergence Education Center, Dongguk University, Korea. Since March 2017, he is with the Department of Game and Multimedia Engineering, Korea Polytechnic University, Korea as an assistant professor. His research interests fall within the general fields of computer science and its applications including distributed computing and algorithms; cloud computing and virtualization; artificial intelligence and big data analytics; mobile and sensor networks; and fault tolerant and resilient techniques.

HeonChang Yu https://orcid.org/0000-0003-2216-595X

He received the B.S., M.S., and Ph.D. degrees in computer science and engineering from Korea University, Seoul, Korea, in 1989, 1991, and 1994, respectively. He has been a Professor of computer science and engineering with Korea University since 1998. From February 2011 to January 2012, he was a Visiting Professor of electrical and computer engineering in Virginia Tech. Since 2015, he has been the Vice President of Korea Information Processing Society, Korea. He was awarded the Okawa Foundation Research Grant of Japan in 2008. His research interests include cloud computing, virtualization, distributed computing, and fault-tolerant systems.

Joon-Min Gil https://orcid.org/0000-0001-6774-8476

He received his B.S. and M.S. degrees in Computer Science from Korea University, Korea in 1994 and 1996, respectively. He received his Ph.D. degree in Computer Science and Engineering from Korea University, Korea in 2000. Before joining in School of Information Technology Engineering, Daegu Catholic University, Dr. Gil was a Senior Researcher in Supercomputing Center, Korea Institute of Science and Technology Information (KISTI), Korea from October 2002 to February 2006. From June 2001 to May 2002, he was a Visiting Research Associate in the Department of Computer Science at the University of Illinois at Chicago, USA. His recent research interests include cloud computing, big data computing, artificial intelligence, distributed computing, and wireless sensor networks.