HMGOWM: A Hybrid Decision Mechanism for Automating Migration of Virtual Machines

Ronghui Cao, Zhuo Tang, *Member, IEEE*, Kenli Li, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

Abstract—Large-scale data centers have been widely used for cloud services, and the stability of various cloud services has received additional attention from users. Although service disruptions are not as catastrophic as they once were, their impact might be more extensive than before. These outages may trigger the migration of virtual machines (VMs) located in the failure node. However, the access time of each VM is random, unlike the accident time, which can be predicted. This means that traditional migration caused by service interruptions may result in a large number of unwanted migrations, regardless of the user's downtime experience. Migration is an expensive process in terms of the resources needed as well as the degradation of application performance during migration. A balance between the recovery time of the service (to minimize the migration resulting from a given placement) and the downtime experience of the users (to minimize the impact of access interruptions) is needed. In this paper, we propose HMGOWM, a hybrid decision-making mechanism for automating the migration of VMs. Our proposed mechanism extends the original VM migration performance cost model, greatly reducing the downtime experience of the users. To achieve high performance and a good load balance, a multi-objective monitoring system for both VMs and physical machine nodes and an adaptive VM migration-scheduling scheme for the OpenStack cloud platform are proposed. Extensive experiment results indicate that the downtime experienced by users can be efficiently reduced and that the implementation of HMGOWM outperforms the original scheduling of the OpenStack cloud platform.

Index Terms—Analytic hierarchy process, cloud computing, live migration, virtualization

1 Introduction

1.1 Motivation

Data centers have become the backbone of the modern economy, from server rooms that power small to medium-sized organizations, to server farms that support major companies and corporations and hyper-scale enterprise data centers that provide cloud-computing services hosted by companies such as Amazon, Facebook, and Google [18]. These data centers rely on massive hardware infrastructures and complex management tasks (e.g., a large number of software upgrades) that can exhibit failures, which, if not handled correctly, can lead to severe implications [11]. During past outages, failures often cascaded to other healthy clusters, dependent services failed, manual mistake-prone recovery code had to be quickly developed during the outage, and users were frustrated and angry [2], [9], [29]. In February of 2003, a number of websites became unavailable after Amazon's website hosting service went down unexpectedly, and it didn't quite break the Internet, but a 4-hour outage at Amazon's AWS cloud computing division caused headaches for hundreds of thousands of websites across the United States [7]. Coincidentally, on Thursday 11 August 2016, 21 percent of Google App Engine applications hosted in the US-CENTRAL region experienced error rates in excess of 10 percent and elevated latency between 13:13 and 15:00 PDT [17]. The stability and user satisfaction of cloud services have become an increasingly significant standard of the total quality of service (QoS) of current and future cloud platforms [28].

It is possible to dramatically reduce the probability of cloud service disruption and the delay of service interruption through the appropriate designs of VM migration strategies for cloud platforms. Live migration is a extremely powerful tool for cluster administrators, allowing separation of hardware and software considerations, and consolidating clustered hardware into a single coherent management domain. If a physical machine needs to be removed from service an administrator may migrate OS instances including the application that they are running to alternative machine(s), freeing the original machine for maintenance [6]. In these situations the combination of virtualization and migration significantly improves manageability. The majority of the studies of VM migration fall into two categories. Some migration mechanisms can effectively achieve multiple requirements, such as energy consumption reduction or resource utilization improvement [8]. Some optimized migration architectures were helpful in reducing the migration time, especially the VM downtime during migration [5], [31].

- *R. Cao, Z. Tang, and K. Li are with the College of Information Science and Engineering, Hunan University, Changsha, Hunan 410006, China, and also with the National Supercomputing Center in Changsha, Changsha, Hunan 410082, China. E-mail: {caoronghui, ztang}@hnu.edu.cn, lkl510@263.net.*
- *K. Li is with the College of Information Science and Engineering, Hunan University, Changsha, Hunan 410006, China, and the National Supercomputing Center in Changsha, Changsha, Hunan 410082, China, and also with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA. E-mail: lik@newpaltz.edu.*

Manuscript received 20 Nov. 2017; revised 16 Aug. 2018; accepted 17 Sept. 2018. Date of publication 4 Oct. 2018; date of current version 8 Oct. 2021. (Corresponding author: Zhuo Tang.)
Digital Object Identifier no. 10.1109/TSC.2018.2873694

1.2 Related Work

In this section, we review three areas related to our study: virtual machine deployment solution, live migration mechanism research, and memory pre-replication optimization.

1.2.1 Placing Virtual Machine

Many researchers have shown that VM placement has always been viewed as a bin-packing problem [36]. A common solution to this problem is the use of a local preferential search method to find a feasible solution; the advantage is a polynomial-time reduction, but only an approximate solution can be obtained. In [39], the authors proposed a system that automates the task of monitoring and detects hot spots, determining a new mapping of physical to virtual resources and initiating the necessary migrations. In [13], the authors proposed a resource manager for homogeneous clusters which performs dynamic consolidation based on constraint programming and takes migration overhead into account. The use of constraint programming allows Entropy to find mappings of tasks to nodes that are better than those found by heuristics based on local optimizations and that are frequently globally optimal in the number of nodes. In [38], the authors proposed an automatic performance tuning strategy to balance the workload in the virtualized cluster system. The disadvantage of this strategy was not considered the full use of computing resources and the VMs were divided into multiple nodes on a balanced basis and each node may exist idle resources. However, along with the rapid development of hardware and network technology in cloud platforms, various problems during the execution of VMs were already unable to be solved by a single resource placement method.

1.2.2 Deploying Live Migration Mechanism

As an important management method for data centers, live migration allows a VM to migrate seamlessly between different physical nodes. Thus, it is widely used for load balancing, system tolerance, energy consumption management, and other application scenarios [12], [34]. In [22], the authors presented a scheduling strategy on load balancing of VM resources based on genetic algorithm and addressed the problem of load imbalance and high migration cost by traditional algorithms after scheduling. In [32], the authors contributed an automatic and transparent mechanism for proactive FT for arbitrary MPI applications. It leverages virtualization techniques combined with health monitoring and load-based migration. In [4], the authors presented a decentralized architecture of the resource management system for Cloud data centers and propose the development of the following policies for continuous optimization of VM placement. In [26], the authors presented a distributed snapshot capability for virtual distributed environments, based on virtual networking system called VIOLIN [25]. However, these models are only applied to cluster performance optimization in cloud platform and different from service applications in cloud platform, since there is no need to consider the VM usage experience from users. In addition, these traditional migration mechanisms not only neglect the service quality of VMs but also easily results in additional problems such as excessive network traffic.

1.2.3 Optimizing Memory Pre-replication

The existing online migration technology primarily uses the memory pre-replication method, and the foremost problem is the large amount of memory data that must be transferred throughout the migration, causing long delays. In [14], the authors implemented and evaluated another strategy for live VM migration called post-copy that defers the memory transfer phase until after the VM's CPU state has already been transferred to the target and resumed there. In [23], the authors presented a novel iterative pre-copy algorithm designed for convergence instead of guaranteeing maximum downtime. In [30], the authors described the design and implementation of a novel approach CR/TR-Motion that adopts checkpointing/recovery and trace/replay technology to provide fast, transparent VM migration which replaces the dirty page with the execution trace file in each round. However, how to reduce the migration delay of VMs while keeping the original platform migration mechanism is still an unresolved issue.

1.3 Our Contributions

It is clear that virtual resource migration and management of cloud platforms are critical research issues for the QoS of cloud services. However, although there has been a significant effort made by many researchers, the majority of the VM scheduling and migration optimization algorithms do not consider the downtime experience of users, especially during migration. In contrast, we focus on the access habits of users before and during migration and propose a hybrid decision mechanism that can effectively reduce the downtime experience and enhance customer satisfaction. Our contributions in this paper, as well as the organization of the paper, are summarized as follows:

- We present a migration downtime comparison of two VMs with workloads based on a VM migration performance model [31].
- We improve the original autoregressive model (*AR (p)*) and design a new order determination method *IF-test* to predict the user behavior.
- We propose a user behavior prediction (UBP) model that combines real-time performance monitoring data and the improved autoregressive model to choice the migrated VM.
- We propose a multi-objective monitoring system on the OpenStack cloud platform [15] to collect the performance data of both VMs and physical nodes".
- By introducing the analytic hierarchy process (AHP) [10], we create a hybrid multi-goal optimization weight method (HMGOWM), which combines user behavior prediction and a hybrid decision mechanism to effectively reduce the downtime experience of users and balance the load of the PM nodes.

To the best of our knowledge, this is the first paper that analytically and comprehensively studies the downtime experience of users during VM migration on cloud platforms. The remainder of this paper is organized as follows: Section 2 discusses related works, while some theoretical background of migration and AHP are introduced in Section 3. Section 4 presents the optimized migration performance cost model, and Section 5 introduces our hybrid decision mechanism.

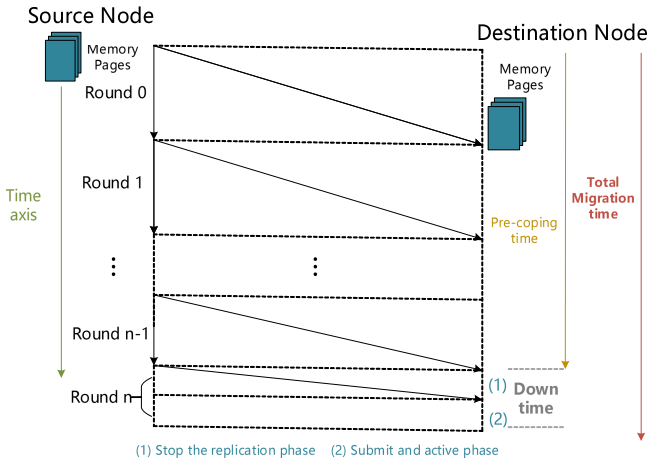


Fig. 1. The process of the pre-copying.

The evaluation of performance of our architecture is presented in Section 6. Finally, the paper is concluded in Section 7 with a preview of future planned work.

2 Theoretical Background

2.1 Memory Pre-copying Approach

Pre-copying [6], [33] is the main implementation of a VM live migration. This method was originally used to migrate the process's address space [37] and is now widely used for physical memory data transmission during live migration [3]. As shown in Fig. 1, to reduce the downtime of migration, the pre-copying phase begins when the VMs are still executing at the source nodes. The source node sends the entire memory image to the destination node, and a portion of the memory pages will be modified during this process; these pages will be transferred in next round. The transferred memory pages of each round are the modified portion of the previous round (i.e., the transmission memory page in round n is the modified page in round $n-1$) until the number of modified memory pages is relatively small or the cycle number reaches a certain threshold.

2.2 Migration Performance Model

Live migration performance modeling involves several factors, including the VM memory size, workload characteristics (denoting the memory-dirtying rate), network transmission rate and migration algorithm. The largest challenge is correctly characterizing the memory access pattern on each excuting workload. In [31], the authors proposed a migration performance model that predicts the cost before the migration and provides decision support to the VM selection algorithm.

Live migration achieves negligible application downtime by iteratively pre-copying the pages dirtied during the previous round of transmission. Assuming the pre-copying algorithm proceeds in n rounds and denoting the data volume transmitted at each round as $V_i (0 \leq i \leq n)$ and the elapsed time at each transferring round as $T_i (0 \leq i \leq n)$, V_0 is equivalent to the VM memory size V_{mem}. The data transmitted in round i can be calculated as follows:

$$V_i = \begin{cases} V_{mem} & i = 0; \\ D_p \cdot T_{i-1} & 0 < i \leq n. \end{cases} \tag{1}$$

Considering the situation where the memory-dirtying rate is smaller than the memory-transmission speed on average, the authors defined the variable λ to represent the ratio of the memory-dirtying rate D_p and the memory transmission rate R

$$\lambda = \frac{D_p}{R}. \tag{2}$$

Then, the total migration latency is

$$T_{mig} = \frac{\sum_{i=0}^{n} V_i}{R} = \frac{V_{mem}}{R} \cdot \frac{1-\lambda^{i+1}}{1-\lambda}. \tag{3}$$

Because T_{mig} is the migration duration that has a negative effect on the performance of executing applications, it is a key performance metric for the migration decision.

2.3 Analytic Hierarchy Process

The AHP is one of the most popular multi-criteria decision-making (MCDM) methods [35]. To the best of our knowledge, it is the only MCDM method that organizes the critical aspects of a problem into a hierarchical structure and simplifies the decision process. It is the only method that is able to measure the consistency of the decision makers' judgments. Additionally, decision makers often prefer to make a decision based on a pairwise comparison that allows them to be more actively involved in the decision-making process.

3 Migration Performance Cost Model

For a given VM, the memory image size allocated by the compute nodes and the memory transmission rate among nodes is fixed. Consequently, the iterative pre-copying would converge faster if λ is smaller. We therefore refer to λ as the convergence coefficient of live VM migration. Additionally, we can conclude that a VM with a smaller memory image and value of λ will help to reduce both network transmission overhead and downtime when migrating. In a cloud environment, in order to maintain the physical machine (PM) nodes running in a relatively stable environment and avoid higher failure rate, the resource of the PM nodes has its own stability threshold which is the upper limit of stable utilization of resource. If some nodes exceed the established threshold, the platform must perform some appropriate VM migration scheduling and selects VMs with relatively small migration times to migrate. Combined with the migration performance cost model, we define the variable γ as the total migration latency ratio of two VMs

$$\gamma = \frac{T_{mig1}}{T_{mig2}} = \frac{V_{mem1}}{V_{mem2}} \cdot \frac{(1-\lambda_1^{n+1}) \cdot (1-\lambda_2)}{(1-\lambda_2^{n+1}) \cdot (1-\lambda_1)}. \tag{4}$$

Ultimately, the value of γ is determined by two factors, the memory image ratio and the fraction of variable λ. As mentioned above, the memory image ratio is fixed and does not change over time. To study the fraction of λ, we build a real migration environment based on the open source OpenStack cloud platform. We choose KVM [27] as the hypervisor and measure the value of λ with different VM configurations and different workloads.

We conduct experiments on 4 enterprise-class PMs with two quad-core Intel Xeon E5420 2.40 GHz processors, 12 GB

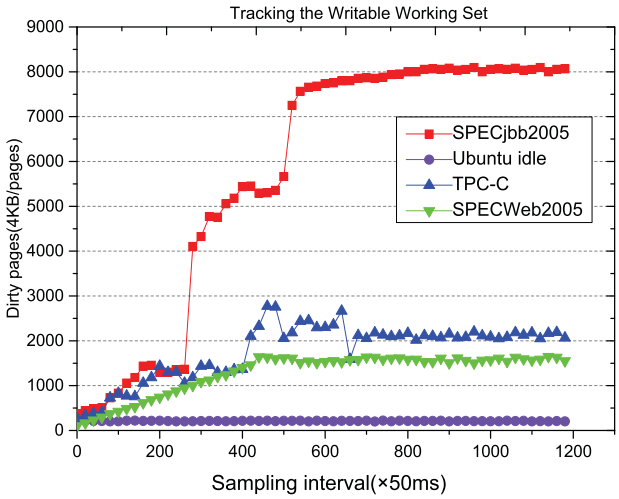


Fig. 2. Writeable working set measurement for different workloads in a VM with 1 GB RAM.

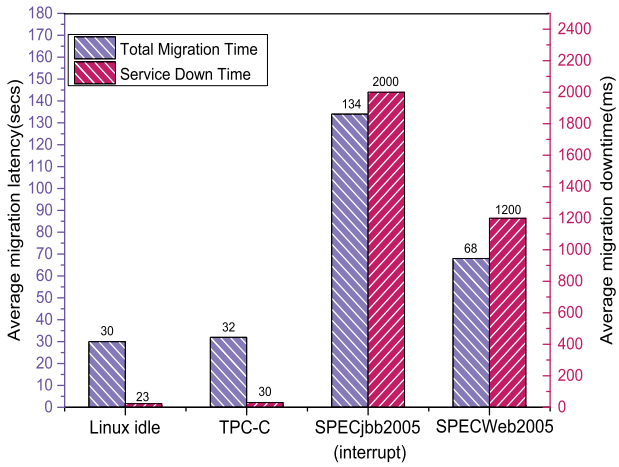


Fig. 3. Migration latency estimation and migration downtime estimation.

memory, one 500 GB SATA hard disk, and two 1 Gbit Ethernet interfaces. The host machines were executing Ubuntu 14.04.3 distribution and the release of OpenStack is Kilo. The VMs were also executing Ubuntu 14.04.3 with the Linux 2.6.20 kernel. In order to simplify the problem, all VMs were configured with two VCPUs and 1 GB RAM. The experiments use four workloads, representative of typical individual applications in the current cloud platform.

(1) SPECjbb2005 [19]: This is a SPEC benchmark for evaluating the performance of server-side Java. SPECjbb2005 provides a new enhanced workload, implemented in a more object-oriented manner to reflect how real-world applications are designed and introduces new features such as XML processing and big decimal computations to allow the benchmark to be a more realistic reflection of current applications.
(2) Ubuntu idle: This is an idle Ubuntu OS for daily use. This workload is used for comparison.
(3) TPC-C [21]: This is an on-line transaction processing benchmark that simulates a complete computing environment where a population of users executes transactions against a database.
(4) SPECweb2005 [20]: This is a next-generation SPEC benchmark for evaluating the performance of World Wide Web servers and includes many sophisticated and state-of-the-art enhancements to meet the modern demands of Web users. We use one client machine with 800 concurrent connections to generate the load for the web server.

The shadow page table [27] of KVM allows us to track the pages to be dirtied within any time window. We initiate different applications in one VM and review the dirty bitmap every 50 ms. Without cleaning the dirty bitmap, we observe the writable working set over a relatively long time window (60s). The number of dirty pages generated by different workloads is shown in Fig. 2.

The slope of the line that joins the origin and the point on each curve represents the memory-dirtying rate. Most applications displayed a higher memory-dirtying rate at the beginning of the measurement, and as the observation time increases, the dirtying rate drops in all applications. This implies that we can measure the writable working set (WWS) [6] when a curve slope at a particular point does not change. The majority of the dirty pages of the on-line transaction processing (OLTP) benchmark are generated by allocating new memory pages for upcoming connections, and only a few pages are repetitively updated. As SPECjbb is a CPU-and-memory-intensive workload with a quite large WWS value and a high memory-dirtying rate, it easily leads to migration failure when executing in VMs. To to avoid long migration latency, this type of VM should be removed from the candidates of migrated VMs.

Fig. 3 shows both latency and downtime of migration under different workloads. To avoid VM migration failure caused by the benchmarking tool SPECjbb2005, we manually set the tool to be closed during migration. SPECjbb2005 and SPECWeb2005 exhibit much longer migration downtime than the other workloads because of their high memory-dirtying rate. In general, both latency and downtime of migration under different workloads show enormous differences based on the different configurations of VMs and the load characteristics of the platforms.

Usually, in order to reduce the cloud service disruption probability, the cloud platform choose some VMs to migrate to a safer location in the presence of (in anticipation of) a region failure or degradation. Whatever the migration cause, the VMs that have smaller (i.e., the memory-dirtying rate is smaller than the memory-transmission rate on average) are good candidates for migration in most case. In fact, after excluding those VMs whose λ are too large, the migration overhead (i.e., the generation velocity of memory dirty pages) is almost the same as when the value of λ is less than 1. For most ordinary users, the λ of two migrated VMs is approximately equal to the memory image ratio, and the the value of V_{mem} is determined by the users' requirements when tenants apply for VMs.

However, when the VM is triggered to migrate, the tenants are temporarily unable to operate the VM that still executes on the source node with no new operation request from the tenant. Therefore, for the tenants, the downtime of migration is the VM total migration time. Thus, the original migration performance (OMP) model did not consider the downtime experience of users. Customer satisfaction acts as the primary motivation for constructing our mechanism,

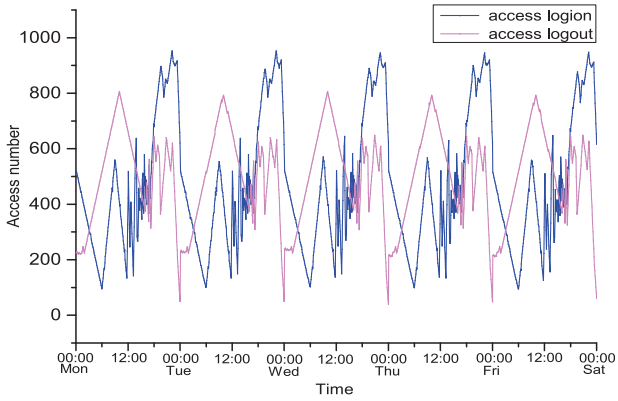


Fig. 4. Distribution diagram of access login and logout.

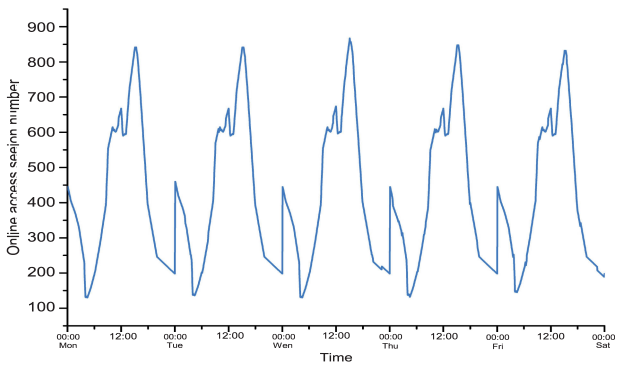


Fig. 5. Distribution diagram of online access session.

the HMGOWM, a hybrid decision mechanism for automating migration.

4 User Access Time Span Analysis

From the above analysis, we found that the access to a VM can be interrupted by the VM migration. Just like the network user session behavior, the entire VM access process can also be divided into three parts, access logion, access logout and access session. The VM access is the most basic tenant behavior and it is meaningful for VM sales customization, resources management and platform maintenance. Additionally, the VM access quality can directly affects the tenant's user experience. However, existing VM migration schemes do not seem to take this factor into account. In most cloud platforms, server failures are affected by various factors, so they occur randomly and resulting in large-scale nonselective VM migrations. In this section, we analyse the tenant access session behavior and cloud platform failure period, and try to find the relationship in detail by describing the users' downtime experience during migration.

In order to analysis the tenant access session behavior, we parse a total of 1000 VMs access processes in the Aliyun platform in December 2016. We split the time of one day into 288 time spans, then calculate the number of access logion and access logout in each time span. The number of access session in four weeks has been averaged statistically, and we get the distribution diagram of access number of VM logion and logout. The distribution diagram is shown in Fig. 4.

Fig. 5 draws the number of online access session of one week on five-minutes basis. It can be seen from Figs. 4 and 5 that the distributions of access session number in working

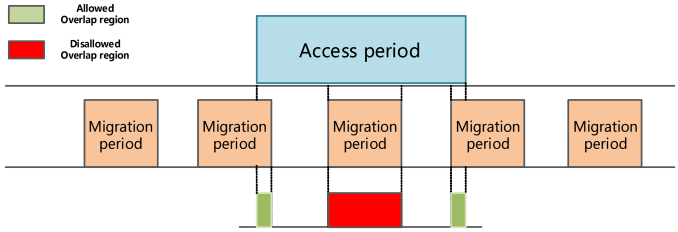


Fig. 6. The relationship between access-period and migration-period of VM.

days are similar and it is quite conceivable that the access period and logion-logout time are periodic fluctuated. Meanwhile, we also collect some global server failures in 2016, from various cloud platforms such as Google cloud platform, AWS, Microsoft Azure, Aliyun and so on. The results show that server failures almost always lead to large-scale random VM migrations but the occurrence time and duration of these failures are randomness. However, the VMs which are being accessed or are going to be accessed do not seem to have been treated differently during migration. Excessive overlaps between the migration-period and access-period of VMs must lead to declines in the user experience. Hence, it is important to reduce the period conflict between VM access and migration. After the platform failure happen, some VMs need to migrate to the alternate server. In the process of migration, tenants may access or are going to access these VMs, the relationship between access-period and migration-period is shown in Fig. 6.

From the above drawing, there are mainly three types of relationships between access-period and migration-period, no overlap region, allowed overlap region and disallowed overlap region. No overlap region, as the name suggests, has no overlapping part between access-period and migration-period. That means the service which is deployed on the VMs and provided by the VMs would not be interrupted by the VM migration. Based on these two factors, if no overlap region, the cloud platform can migrate VMs without affecting the access session of users, so the user experience will not be influenced by the migration. The two other relationships, both have overlap areas between two periods, and the difference is the overlapping area range. In the allowed overlap region, only few overlap areas and the region upper limit has been determined by the tolerance of users' downtime experience. For disallowed overlap region, the migration period is completely included in the access region. That means users are bound to experience the whole migration process which causes a longer downtime experience during access session. So we need to reduce the probability of this situation or even avoid the disallowed overlap region. To improve the user access experience during migration, the HMGOWM needs to predict the time of both access login and logout, and calculates the whole migration time based on Eq. (3). Further more, different overlap regions is match with different migration weights which help to determine the ultimate migrated VM and reduce the coincident factor between migration-period and access-period. For that, the time of login and logout of access region must be predicted to calculate the coincident factor between two periods with different migration weights.

5 Hybrid Decision Mechanism

To solve the problems discussed previously, we have carefully designed the HMGOWM. The main goal of our proposed mechanism is to reduce the downtime experience of users during VM migration by predicting the user access habits while simplifying the VM migration schedule model. To to accomplish this goal, there are several steps:

- Design a multi-objective monitoring system on cloud platforms that monitors and collects resources from both PM nodes and VMs.
- Predict the user behavior (VM access period and memory utilization during migration) based on our improved order determination method *IF-test*.
- Implement a VM migration mechanism based on AHP.

5.1 Multi-objective Monitoring System on Cloud Platforms

5.1.1 Cluster Monitoring

There are common methods to monitor and collect various resource data from the PM nodes. One method is to use the Linux command line to query the usage data and record of resources, and the other method is to read the content of the */proc* file directly. For the firs method, we should view the resource state by executing the Linux Shell command and analyze the returned information to obtain the required performance data. However, this requires the user to be familiar with how to deconstruct the returned information and retrieve useful data, and the administrator must be well acquainted with the feedback information structure returned by common Linux commands. At the same time, it takes an excessive amount of time to obtain the performance data of each PM node, and the system load of the frequent instruction input cannot be overlooked.

/proc is a pseudo filesystem where users and applications can access the relevant execution state information through various sub-files or sub-directories under the */proc* directory and can change some parameters in the Linux kernel. It is different from traditional files that are stored on a disk because Linux stores */proc* files in memory to record relative parameters produced during execution of the running process. It is more efficient to obtain the monitoring data by reading the content of the */proc* files, so we use this method to collect the status information from the PM nodes.

5.1.2 Virtual Resources Monitoring

To collect data from the VMs, we use the open source project *Collectd*, a daemon that collects system and application performance metrics periodically and provides mechanisms to store the values in a variety of manners [24]. It gathers metrics from various sources such as the operating system, applications, log files and external devices, and stores this information or makes it available over the network. These statistics can be used for system monitoring, finding the performance bottlenecks (i.e., performance analysis) and predicting the system load (i.e., capacity planning). It is written in the *C* programming language for performance and portability, allowing it to execute on systems without a scripting language or cron daemon, such as embedded systems. It also includes optimizations and features to handle hundreds of thousands of metrics. The daemon provides over 100 plugins that range from standard cases to very specialized and advanced topics. It provides powerful networking features and is extensible in numerous ways. The monitoring data flow of the resource status of cloud platforms is shown in Fig. 7.

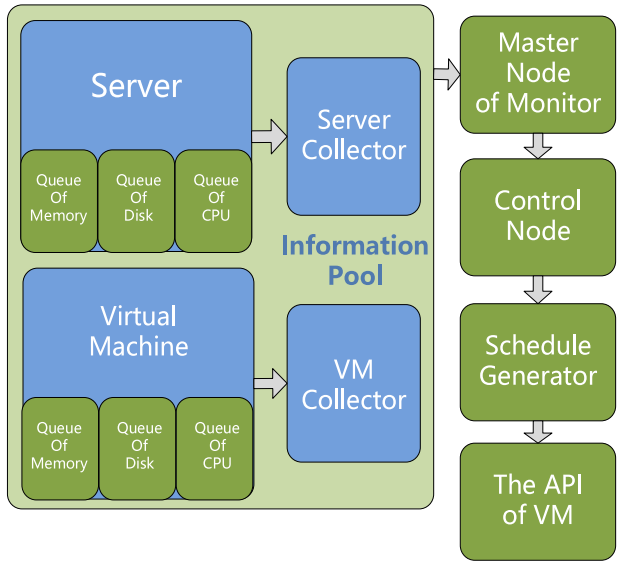


Fig. 7. Resource state monitor data flow.

The detailed steps are not described within the scope of this paper. The virtualization tool *libvirt* is deployed on the PM operating systems, so it reduces the effect of the performance of the monitoring system. To some extent, it also enhances the user experience on the VMs.

The master node collects the configuration information of both the VMs and the servers from the resource allocation manager. It then provides the feedback information to the *Nova* component (a cloud computing fabric controller in the OpenStack cloud platform that supporting a wide variety of computer technologies including libvirt, Hyper-V, VMware, XenServer and OpenStack Ironic [16]) that are under the same control node. The collected information from the VMs and the PM nodes pass the VM scheduler and are stored in the information pool that consists of a series of resource message queues. In this paper, we primarily monitor three utilization indicators: the CPU, the memory and the disk. Each server corresponds to three queues that are used to save the utilization of the CPU, memory and disk on the PMs. To save memory space, only the latest status information in the scheduling queues is useful.

5.2 User Behavior Prediction

After analyzing the VM access period and VM's memory usage under different workloads, we find that if the memory utilization and the access frequency is very high within a specific period of time T, the probability that the memory usage and the access period remains the same or increases in the next period T is higher. As a result, to some extent, the memory change and the VM access probability during period T can be predicted by the change in the memory usage and the access period in the previous cycle. Therefore, we introduce the autoregressive prediction model *AR(p)* [1]

to predict the user behavior (memory usage and access probability) during migration. In order to avoid repetitive description, we use memory prediction to expound the optimized prediction model.

We receive a set of discrete memory values by monitoring, and use these data to match the autoregressive model *AR(p)*. The memory value m_t on time t is obtained through the linear combination of memory values $m_{t-1}, m_{t-2}, \ldots, m_{t-p}$ and the white noise of time t. Therefore, the memory prediction value of period t is as follows:

$$m_t = \psi_1 m_{t-1} + \psi_2 m_{t-2} + \cdots + \psi_p m_{t-p} + \xi_t \quad t = 0, \pm 1, \pm 2, \ldots. \tag{5}$$

We assume that the time series parameters are certain and consider the parameter estimation. According to the least square estimation method of the linear model, the system of linear equations can be converted into the following three vectors:

$$Y = \begin{bmatrix} m_{p+1} \\ m_{p+2} \\ \vdots \\ m_n \end{bmatrix},$$

$$X = \begin{bmatrix} m_p & m_{p-1} & \cdots & m_1 \\ m_{p+1} & m_p & \cdots & m_2 \\ \vdots & \vdots & \ddots & \vdots \\ m_{n-1} & m_{n-2} & \cdots & m_{n-p} \end{bmatrix},$$

$$\xi = \begin{bmatrix} \xi_{p+1} \\ \xi_{p+2} \\ \vdots \\ \xi_p \end{bmatrix}.$$

Therefore, we can rewrite the above equation as $Y = X\psi + \xi$, so the linear least squares estimation of ψ can be expressed as follows:

$$\hat{\psi} = (X'X)^{-1}X'Y. \tag{6}$$

In the cloud environment, because the memory monitoring data sample is larger, we introduce the *F-test* order method to determine the order. We use the observation data of sample length N to fit the $AR(p)$ model and find the residual sum of squares is $RSS = S(p)$. Therefore, the simplest way to determine the order is to establish a series of $AR(p)$ models with $p = 1, 2, \ldots$ and find the residual function $S(p)$. Assuming S_0 is the residual sum of squares of the one-dimensional $AR(p)$ and S_1 is the residual sum of squares of the one-dimensional $AR(p+1)$, the statistic is as follows:

$$F = \frac{S_0 - S_1}{S_0} \cdot (N - p), \tag{7}$$

where the statistic follows the $F(1, (N-p))$ distribution.

Under the given significant level ξ, we can check the *F*-distribution table to obtain the critical value of F_ξ. When the *F-test* is not significant, such as $F < F_\xi$, the order is assumed to be applicable. Otherwise, the order of the $AR(p)$ model is not applicable and must be increased. To determine the order, we generally increase the order gradually from the $AR(1)$ model, using cyclic examination until it is accepted.

TABLE 1
The Memory Prediction Modeling of VMs During the Migration

$AR(p)$	The order of model						RSS	F	$F_{\xi(1,N-p)}$
	$\widehat{\psi_1}$	$\widehat{\psi_2}$	$\widehat{\psi_3}$	$\widehat{\psi_4}$	$\widehat{\psi_5}$	$\widehat{\psi_6}$			
AR(1)	1.03	-	-	-	-	-	455	-	-
AR(2)	1.96	-0.95	-	-	-	-	112	8.2	4.26
AR(3)	2.20	-1.65	0.31	-	-	-	105	1.9	4.25
AR(4)	2.33	-1.52	0.46	-0.05	-	-	91	6.5	4.24
AR(5)	1.98	-1.26	-0.06	0.43	-0.27	-	73	2.6	4.32
AR(6)	1.06	-1.28	-0.08	0.13	0.29	-0.45	69	1.7	4.33

However, the sample randomness easily leads to another case where the appropriate model is $AR(p^*)$, but the obtained model is $(p^* - 2)$. This is because although the difference between $AR(p^*)$ and $AR(p^* - 1)$ is very significant, the difference between $AR(p^* - 1)$ and $AR(p^* - 2)$ is probably not noticeable because of the sample randomness.

Referring to the scenario mentioned above, we design an improved order determination method *IF-test* that adjusts the first inspection from $AR(1)$ to $AR(p_0)$, and the first inspection does not determine the appropriate order p^* but judges the direction of p^*. Similarly, under the given significance level of ξ, by checking the distribution table of F, we can obtain the critical value of F_ξ. In the case of $F \geq F_\xi$, the appropriate order should be higher than p_0, and the order can be determined by a traditional method. When $F < F_\xi$, the order is less than or equal to p_0, and the order is determined by the successive descending order, i.e., first fitting the $AR(p-1)$ model and then testing $AR(p)$ and $AR(p-1)$ for the existence of significant differences. The test statistic of the *IF-test* is as follows:

$$F' = \frac{(RSS(p-1) - RSS(p))(N - p + 1)}{RSS(p)}, \tag{8}$$

where the statistic follows the $F'(1, (N-p+1))$ distribution.

We can also check the distribution table of F' to obtain the critical value F'_ξ under the given significance level of ξ. When $F' \geq F'_\xi$, the $AR(P)$ model is considered to be suitable. Otherwise, the model is inappropriate. Fitting the model $AR(p_0 - 2)$ and testing $AR(p_0 - 1)$ and $AR(p_0 - 2)$, we can eventually determine the correct order.

To test the superiority of the *IF-test*, we prepare PMs with the same resource configuration. Depending on the resource type, we divide the VMs into three categories: the CPU usage, the memory usage and the disk usage. We select 30 consecutive memory monitoring data points from each VM to fit the $AR(p)$ model. Under the given significant ξ, integrating three types of modeling data, the modeling process is shown in Table 1.

Based on the above information and compared to the actual values, the traditional homogeneity of the *F-test* variance determines that the order is 2. The maximum errors in the three types of VMs are 8.78, 8.81 and 8.88 percent, and the residual sums of squares are 112.08, 115.9 and 113.9. However, in the improved homogeneity of the variance

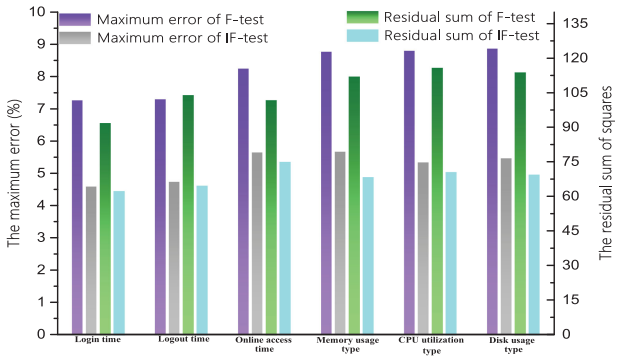


Fig. 8. The comparison of maximum errors and residual sums of squares between *F-test* and *IF-test*.

method *IF-test*, the order is 6. The maximum errors are 5.69, 5.36 and 5.49 percent, and the residual sums of squares of the *F-test* are 68.56, 70.78 and 69.69.

In respect of access period prediction, we also receive a set of access session data by monitoring and select 30 consecutive access session data from three types of VMs to fit the $AR(p)$ model. Under the given significant ξ, integrating three types of session time data: login time, logout time and online access time. Compared to the actual values of access session, the traditional homogeneity of the *F-test* variance determines that the order is 3. The maximum errors in the three types of access data are 7.28, 7.31 and 8.26 percent, and the residual sums of squares are 92.01, 104.7 and 101.86. Obviously, the order which is calculated by using the improved homogeneity of the variance method *IF-test* is different from the traditional variance, is 5. The maximum errors are 4.61, 4.76 and 5.67 percent , and the residual sums of squares of the *F-test* are 62.54, 64.86 and 75.21.

We conclude that for memory usage prediction, the $AR(6)$ model is better than the $AR(2)$ model and for the user access session prediction, the $AR(5)$ is a better choice. The various types of comparison data are shown in Fig. 8.

5.3 User Behavior Prediction Model

After determining the order, we combine the migration performance cost model, the relationship between two periods with the improved order determination method *IF-test*, and propose a user behaviour prediction model model.

In terms of memory prediction during migration, the initial forecast period is sampled as $\frac{V_{mem}}{R}$, and the initial prediction frequency is sampled as $\frac{V_{mem}}{R \cdot M}$. The parameter R denotes the memory transfer speed during migration and M denotes the monitoring frequency of the memory utilization. Therefore, the UBP should contain two aspects, the migration latency T_{mig} and the prediction value of memory usage. We use M_{mem} to denote the average value of the memory prediction during migration and the index is dynamic. The model can be expressed as the following:

$$T'_{mig} = (T_{mig}, M_{mem}), \tag{9}$$

and

$$M_{mem} = \frac{\sum_{t=t_0}^{N} m_t}{\frac{V_{mem}}{R \cdot M}}, \tag{10}$$

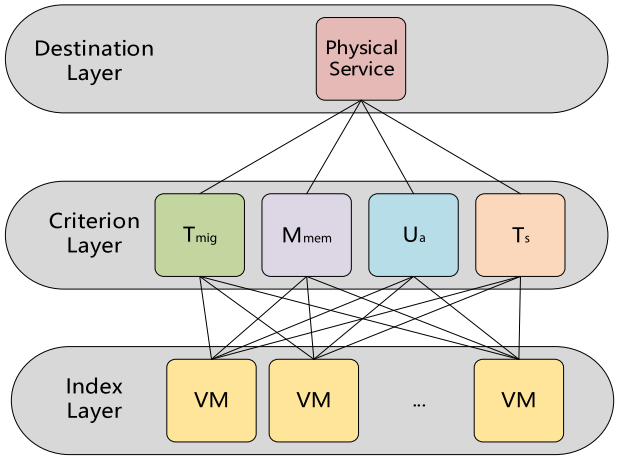


Fig. 9. Hierarchical chart of physical servers.

The parameter t_0 denotes the initial prediction point when the migration begins, and the parameter N denotes the cutoff point that is determined by the prediction frequency $\frac{V_{mem}}{R \cdot M}$. When users apply VMs, the size of V_{mem} is determined by the users' requirements.

On the other hand, as we know, the user experience during migration decreases as the coincidence rate increases between the VM migration period and access period. In other words, the quality of user experience during migration, to some extent, it is equal to the VM migration probability during access period. The access period can be determined by the time of VM login and logout. After determining the order for user access session prediction, we use U_a to represent the coincidence rate of migration period and access period based on the above content:

$$U_a = (P_{mig}, P_{acc}), \tag{11}$$

where P_{mig} denotes the migration period which is calculated by Eq. (3) and P_{acc} denotes the predicted access period.

Compared to the OMP model, the proposed model is based on user behaviour (memory and VM access period) monitoring and prediction and has some advantages. First, the integration of real-time monitoring data provides more diversified and real-time prediction results. Second, the memory change rate to some extent reflects the generation rate of the dirty pages and the memory monitoring data also reflects the operating frequency of the guest OS. The access period prediction also can directly reflects the user behavior of VM access. The relationships between these three variables are directly proportional.

5.4 The Choice of the Source VMs

Next, we use the UBP model to determine the choice of migrated VMs. When specific resource indicators on the PMs exceed the threshold and affect the VM service quality located in these PMs, we must choose some VMs to migrate. This paper discusses only the case where a single service resource exceeds the threshold of the PMs. Combined with the analytic hierarchy process [18], we first establish an index evaluation system for the PMs. The hierarchical chart of the PMs is shown in Fig. 9.

Parameters T_{mig}, U_a and M_{mem} were previously defined. Parameter T_s represents an initial value whose type is the

resource that exceeds the threshold on one PM. When the evaluation system is established and is different from the traditional $AR(p)$ method, the weighting from criterion layer to destination layer can be expressed as follows:

$$\begin{bmatrix} \mathrm{PS} & C_{12} & C_{13} & C_{14} \\ C_{21} & 1 & C_{23} & C_{24} \\ C_{31} & C_{32} & 1 & C_{34} \\ C_{41} & C_{42} & C_{43} & 1 \end{bmatrix}, \tag{12}$$

where C_ij denotes the weight comparison of different elements in criterion layer and PS denotes the physical service in destination layer.

Different from the traditional artificial graded, HMGOWM uses the prediction value of both the memory usage and the access session period during VM migration to construct the judgment matrix. For the index layer, we combine the related influence parameters with the criterion layer and the index layer to generate the following four comparison matrixes

$$\begin{bmatrix} T\_mig & V_{12} & \cdots & V_{1n} \\ V_{21} & 1 & \cdots & V_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ V_{n1} & V_{n2} & \cdots & 1 \end{bmatrix}, \begin{bmatrix} M\_mem & V_{12} & \cdots & V_{1n} \\ V_{21} & 1 & \cdots & V_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ V_{n1} & V_{n2} & \cdots & 1 \end{bmatrix},$$

$$\begin{bmatrix} U\_a & V_{12} & \cdots & V_{1n} \\ V_{21} & 1 & \cdots & V_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ V_{n1} & V_{n2} & \cdots & 1 \end{bmatrix} \text{and} \begin{bmatrix} T\_s & V_{12} & \cdots & V_{1n} \\ V_{21} & 1 & \cdots & V_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ V_{n1} & V_{n2} & \cdots & 1 \end{bmatrix}. \tag{13}$$

where V_{ij} denotes the weight ratio of each VM on the index layer.

According to the analytic hierarchy process, the final weight comparison is the sets of weight that consists of five comparison matrixes which four are the VM comparisons based on the each parameter (T_mig, U_a, M_mem and T_s) in criterion layer and one is the criterion comparison based on the physical server in destination layer.

This new weight comparison above is used to replace the decision made by evaluators to reduce the artificial instability factor. Then, based on the judgment matrix, we calculate the weight value of each VM on unstable PMs that exceed the established threshold. Finally, based on the weight sequencing, the VM with the smallest weight is selected as the source VM to migrate.

5.5 The Choice of Destination Computing Node

Next, we need to choose the destination-computing node. When VMs must be migrated, the monitoring system collects the average usage M_1 of the PMs' resources used by these VMs while they are executing. The vector M_1 also represents the estimated value of these migrated VMs for resource use on the destination node, as follows:

$$M_1 = (C_1, M_1, D_1), \tag{14}$$

where C_1, M_1 and D_1 denote the average usage values of the CPU utilization, memory usage and disk usage of the source VMs, respectively.

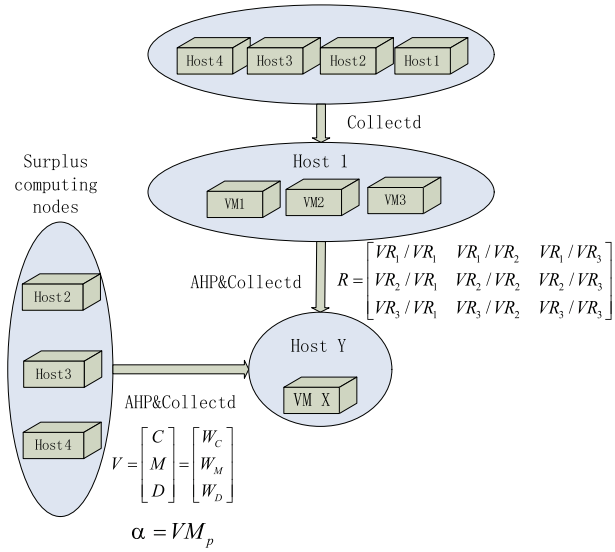


Fig. 10. The process of the decision mechanism.

The monitoring system also collects the current performance monitoring data for all PMs. We denote these data as the vector M_p, as follows:

$$M_p = (C_p, M_p, D_p), \tag{15}$$

where C_p denotes the CPU utilization of the PM_p, M_p denotes the memory usage of the PM, and D_p denotes the disk usage of the PM.

Meanwhile, the destination nodes must satisfy a premise. After migration, the migrated VMs cannot affect other VMs that are providing normal services, i.e., the requirement to meet the following condition:

$$M_p + M_1 < M_{th}, \tag{16}$$

where M_{th} denotes the threshold of each resource on the destination nodes.

After the destination-computing node is preliminarily screened, we must choose the optimal of the remaining servers. When the VM is established, the users record the weight ratio of each resource in the quota. Therefore, we obtain the weight vectors of the following three types of resources:

$$V = \begin{bmatrix} C \\ M \\ D \end{bmatrix} = \begin{bmatrix} W_c \\ W_m \\ W_d \end{bmatrix}. \tag{17}$$

Combining (13) with (14) and (15), we calculate each resource using the information from the remaining compute nodes. The result for the collection is $\alpha = VM_p$. The destination node has the minimum α among the remaining computing nodes. The surplus quantity of various resources is the most suitable for source VMs. In order to reduce resource consumption on server and maintain the accurate prediction for user behaviour, we set the interval of gathering informantion is one week. Furthermore, the real-time matrix and vector are updated in this period to ensure the real-time migration performance. The entire HMGOWM process is shown in Fig. 10.

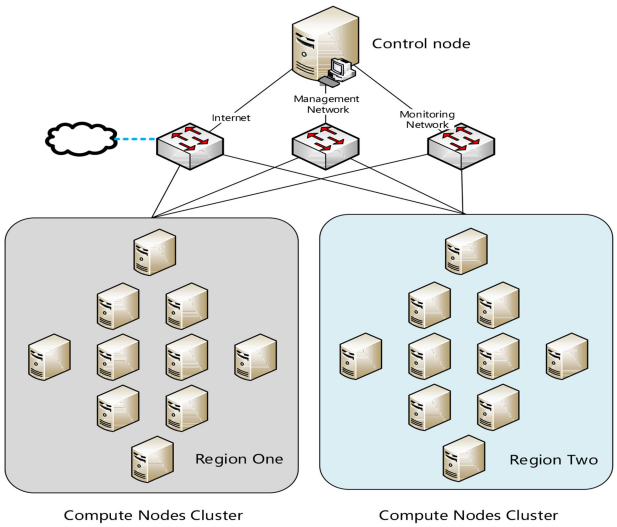


Fig. 11. The topology of network.

6 Experiments and Evaluations

We validate the advantages of the proposed HMGOWM in two experimental settings: 1) user experience experiments in which we are primarily concerned with the access probability of the migrated VM during migration and 2) platform experiments where, for a given set of VMs, we are concerned with the contrast between the traditional VM scheduling of OpenStack and the proposed mechanism.

6.1 Experiment Setup

In this paper, we consider only the least amount of traffic in the overall environment. In the OpenStack cloud platform, the proposed monitoring system and the dynamic scheduling method should be deployed in the same physical environment. We choose one control node and 20 computing nodes to structure the entire cloud environment. Every 10 nodes can constitute a Region in the OpenStack platform. Specifically, each PM is equipped with 2 quad-core Intel Xeon E5420 2.40 GHz processors, 12 GB memory, 500 GB network file system (NFS) storage, and three 1 Gbps Ethernet interfaces used for the NFS storage traffic and VM applications. All the PMs are connected through a 1 Gbps network switch. The PMs are executing on Ubuntu 12.04.2 distribution and the Linux 2.6.20-KVM kernel. The VMs are executing on Ubuntu 12.04.2 with the Linux 2.6.20 kernel. All the VMs can be accessed through NFS storage and the topology of network as shown in the Fig. 11.

6.2 User Experience Experiments

6.2.1 Performance Results Overview

In this section, we evaluate the migration probability of different workloads based on the UBP model and compare the results with the original migration performance model. The results of the comparative analyses are presented in Fig. 12. During the experiment, the variation trend of migration probabilities under different workloads is same as the trend of the number of dirty page under these workloads in OMP model (A VM with a large number of dirty pages may cause higher migration probabilities). This is because for different workloads, the migration probability

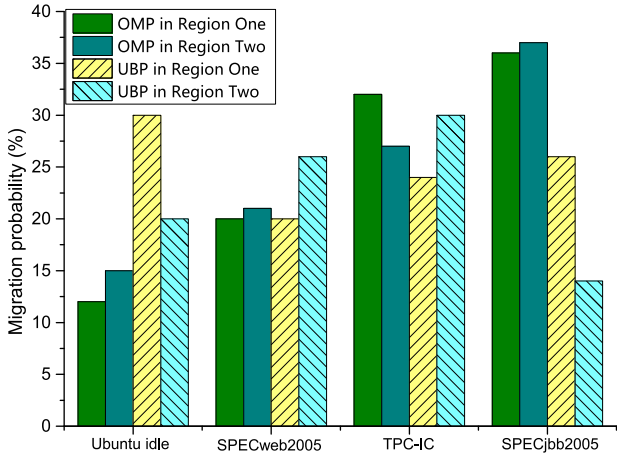


Fig. 12. The migration probability of VMs with different workloads.

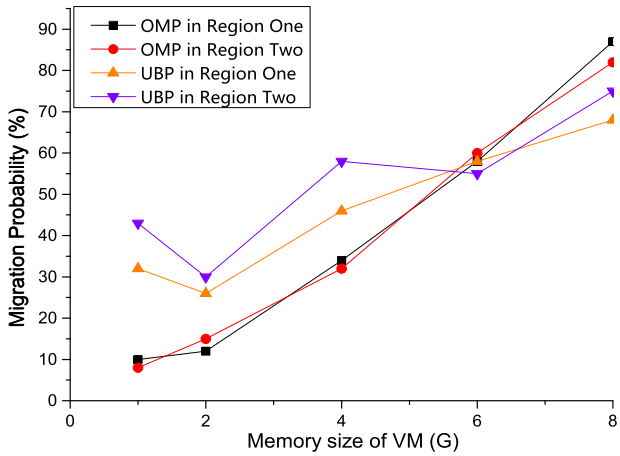


Fig. 13. The migration probability of VMs with different memory sizes.

is approximately λ in the OMP model. However, with the introduction of user behavior in the UBP model, the choice of migrated VMs does not only consider the change rate of the memory dirty pages, but the user experience during migration. The result also shows that in OMP model, the migration probability of VM increases with the number of dirty pages of memory. However, the migration probability in UBP shows a irregular variation with different applications. That means the memory dirty page is not the only determining factor for migration probability, and HMGOWM were not effected by the region partition on the OpenStack cloud platform.

6.2.2 Detailed Results

1) *The Average Migration Probability with a Different VM Memory Size in Each Cluster*: In this experiment, the workload is SPECWeb2005, and each VM is equipped with two VCPUs and a 100GB disk, but the memory size in this group of VMs is incremental. From Fig. 13, it is seen that during the memory increasing process of the VMs, the change of the OMP model is regular and UBP model is irregular. This is because in the case of equal λ, the total migration latency in the OMP model increases linearly with the increase of the VM memory. In the UBP model, it must consider not only the size of the memory but also the memory change

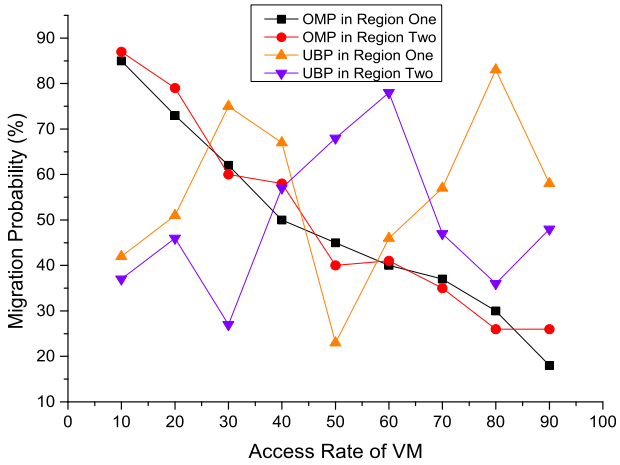


Fig. 14. The migration probability of VMs with different VM access rates.

rate in real-time. Therefore the migration probability is increasingly irregular.

2) *The Average Migration Probability with Different VM Access Rates in Each Cluster*: For the same operation of the VMs with two VCPUs, 2 GB memory and a 100 GB disk, we vary the access rate of the VMs to evaluate the migration probability. The average migration probability for different VM access rates is shown in Fig. 14. In UBP, the migration probability of a certain VM decreases slowly with the increase of the access rate of the VMs. However, it appears that the OMP model does not consider the access habits of the users (i.e., the memory change rate in real-time), and with the increase in the access rate of the VMs, the migration probability may not decrease. During UBP implementation, the weight comparison of the migrated VMs includes not only the memory size but also the average valuation of the memory prediction and a valid resource value hat, as defined in Fig. 6. Therefore, the OMP model easily leads to large-scale service outages during VM migration.

From detailed results we can see clearly that both the number of computing node and region partition in platforms have no impact on the migration probability of VMs. According to the official documents of OpenStack, large-scale deployment supports the multi-region characteristic to provide a better control of computing nodes. Therefore, we believe that HMGOWM can be further implement in a large-scale environment in practice.

6.3 Platform Experiments

6.3.1 Scene Simulation

To give a detailed analysis about the VM migration process on the OpenStack platform and simulate different demands from different tenants on real cloud services, we use one control node and three computing nodes to strcture a small-scale cloud platform. We apply three types of VMs that focus on different resources. The configuration of the CPU-type is dual-core, 2 GB memory and 50 GB disk storage; the configuration of the memory-type is single-core, 4 GB memory and 50 GB storage, and the configuration of the disk-type is single-core, 2 GB memory and 100 GB of disk storage. We create three VMs for each type and order is the CPU type first, the memory type second and the disk type third, for a total of nine VMs in three rounds. Based on the different types of VMs, we artificially establish the resource weight of three VM types when they are created and obtain three types of a judgment matrix, as follows:

TABLE 2
Pairwise Comparison Matrix in AHP Method

S_c	CPU	Memory	Disk	Priorities
CPU utilization	1	2	4	0.5714
Memory Usage	1/2	1	2	0.2857
Disk capacity	1/4	1/2	1	0.1429
S_m	CPU	Memory	Disk	Priorities
CPU utilization	1	1/2	2	0.2857
Memory Usage	2	1	4	0.5714
Disk capacity	1/2	1/4	1	0.1429
S_d	CPU	Memory	Disk	Priorities
CPU utilization	1	1/2	1/4	0.1429
Memory Usage	2	1	1/2	0.2857
Disk capacity	4	2	1	0.5714

$$S_c = \begin{bmatrix} 1 & 2 & 4 \\ 1/2 & 1 & 2 \\ 1/4 & 1/2 & 1 \end{bmatrix},$$

$$S_m = \begin{bmatrix} 1 & 1/2 & 2 \\ 2 & 1 & 4 \\ 1/4 & 1/2 & 1 \end{bmatrix} \text{and } S_d = \begin{bmatrix} 1 & 1/2 & 1/4 \\ 2 & 1 & 1/2 \\ 4 & 2 & 1 \end{bmatrix}.$$

The eigenvector of judgment matrix after normalization is the weight vector of the evaluation unit and the weights of three types of VMs are presented in Table 2.

6.3.2 Migration Procedure Analysis

Before migration, one CPU-type VM and two memory-type VMs were deployed on compute1, one memory-type VM and two disk-type VMs were deployed on compute2, and the remaining three VMs were located on compute3. From the perspective of the default scheduling scheme on OpenStack, the full efficiency of resource utilization is not ideal on a cloud platform. The residual amount of memory on compute1 was very limited, and the PM node compute3 under the same configuration reserved 50 percent of the memory. From the perspective of load balancing of the CPU, the results of the native scheduling program were not very desirable.

After a period of operation on the platform, the deployment of the VMs tends to change, and the memory resource of compute1 exceeds the threshold (we established the resource threshold at 80 percent) resulting in the first migration of a memory-type VM to compute3. However, after this migration, the platform becomes more unreasonable than before. Compute3 with a heavy load soon migrates a VM. The second migration is also caused by the memory resource and leads to a disk-type VM migration from compute3 to compute1. The entire process of change for the three types of resource utilization rates on the PM nodes is shown in Fig. 15.

It can be seen from Fig. 15a that both compute1 and compute3 always exceeded the threshold. After two migrations,

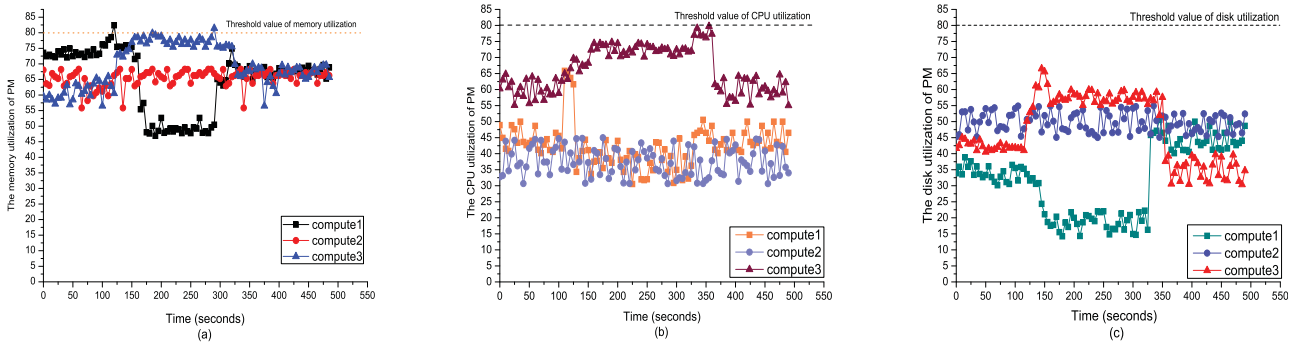


Fig. 15. Three types of resource utilization rate on OpenStack through real-time monitoring during the process of migration. We show the monitoring date of memory utilization(a) exceeds the preset threshold two times. We also show the real-time monitoring date of CPU utilization (b) and disk utilization (c) in the same period.

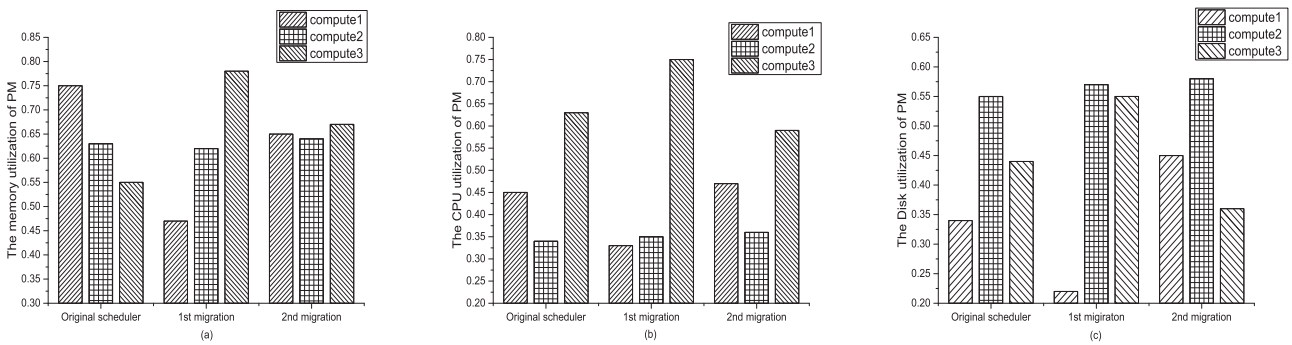


Fig. 16. Performance comparison of original scheduling algorithm on OpenStack and two times of migration of VMs in HMGOWM: (a) under hybrid decision mechanism migration strategy, for achieving memory utilization in OpenStack. We also show other performance in cloud platform with (b) CPU utilization and (c) disk utilization of PM during the process of hybrid decision mechanism.

various resources drop below the threshold, and the memory utilization of each node remains at a medium level. Thus, it can be seen that the memory limit of migration policies can effectively arrange the memory distribution of the cluster. In addition, during the migration, there is no recurrence of the memory exceeding the threshold. For compute2, there is no memory change, and no migration occurred on this node. Therefore, in the entire process of dynamic management, the other VMs are not affected by migration.

Fig. 15b shows the change rate of the CPU utilization of the three nodes during migration. We observe that the CPU utilization increases markedly both times in the migration on compute1 and compute3, but does not exceed the threshold value (we also established the resources threshold at 80 percent). The CPU utilization of compute2 with no migration task remains unaffected. Meanwhile, Fig. 15c shows the change rate of the disk utilization of each node. We observe that OpenStack remained relatively stable in disk resources and was also less affected by migration.

We compare three types of resource utilization with the native scheduler on OpenStack, the first migration and final deployment situation as shown in Fig. 16. We observe that the maximum difference of the average memory utilization ranges up to 20 percent under the native scheduler of OpenStack, but the maximum difference of the hybrid decision mechanism is only 3 percent. However, the maximum difference of the memory utilization under HMGOWM is worse in the first migration. This is because the migration process is divided into multiple migrations; migrating a small number of free VMs at a time is a more acceptable approach to increase the stability of the PMs and the user experience. The current mechanism schedules only one VM when any resource exceeds the preset threshold. We also observe that the deployment of the VMs in the second stage enables the PMs to achieve a better load balance under HMGOWM. After the second migration, the average memory utilization of three nodes maintains a normal level, and the maximum difference between the nodes is less than 6 percent.

Figs. 16b and 16c provide a performance comparison of the CPU and disk utilization with the native scheduler. We compare the effects of load balancing and see that the results of the disk resource are inferior to the memory resource. This is because the entire decision matrix is weighted towards the memory resources among the three resource indexes. Compared to the deployment situation of the first migration, the various resources become more balanced with a stable mechanism environment. The results of these analyses show that the HMGOWM incurs a relatively stable environment of the multidimensional resources for both the PMs and VMs on cloud platforms.

7 Conclusion and Future Work

User experience and satisfaction have become important indicators for cloud platforms. This paper has proposed HMGOWM, which combines memory prediction and a hybrid decision mechanism to reduce the downtime experience of users and to balance the load of the PM nodes. HMGOWM is compatible with the run-time of the original scheduling of VMs on OpenStack. Several measures have been taken to reduce the downtime experience of users during migration, such as designing a multi-objective monitoring

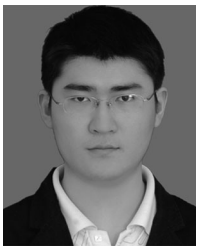
system on cloud platforms, an user behavior prediction model, and an adaptive VM migration-scheduling scheme to achieve load-balance among cloud platforms and reduce the migration probability of VMs with high access probability. Future research will focus on how to achieve this mechanism on different cloud platforms. At the same time, in terms of memory prediction we will consider the more complex situation that application load may fluctuate depending on user behavior, periodic usage patterns, time-zones etc. Additionally, this scenario comparison of an AHP matrix is established automatically based on user demand and our proposed framework will be extended to address more complex user needs in the future.

Acknowledgments

The research was partially funded by the National Outstanding Youth Science Program of National Natural Science Foundation of China (Grant Nos. 6162520), the Program of National Natural Science Foundation of China (Grant Nos. 61751204), the International (Regional) Cooperation and Exchange Program of National Natural Science Foundation of China (Grant No. 61661146006) and the National Key R&D Program of China (Grant Nos. SQ2018YFB020061).

References

[1] H. Akaike, "Fitting autoregressive models for prediction," *Ann. Inst. Statistical Math.*, vol. 21, no. 1, pp. 243–247, 1969.
[2] Summary of the Amazon EC2 and Amazon RDS service disruption in the US east region. [Online]. Avaliable: https://aws.amazon.com/cn/message/65648/, Apr. 2011.
[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *ACM SIGOPS Operating Syst. Rev.*, vol. 37, pp. 164–177, 2003.
[4] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proc. 10th IEEE/ACM Int. Conf. Cluster Cloud Grid Comput.*, 2010, pp. 826–831.
[5] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg, "Live wide-area migration of virtual machines including local persistent state," in *Proc. 3rd Int. Conf. Virtual Execution Environments*, 2007, pp. 169–179.
[6] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proc. 2nd Conf. Symp. Netw. Syst. Des. Implementation-Volume 2*, 2005, pp. 273–286.
[7] Massive Amazon cloud service outage disrupts sites, 2017. [Online]. Available: https://www.oathkeepers.org/massive-amazon-cloud-service-outage-disrupts-sites/
[8] C. Ghribi, M. Hadji, and D. Zeghlache, "Energy efficient VM scheduling for cloud data centers: Exact allocation and migration algorithms," in *Proc. 13th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, 2013, pp. 671–678.
[9] GmailBlog, "More on todays gmail issue," Feb. 2009. [Online]. Available: http://gmailblog.blogspot.com/2009/02/update-on-todays-gmail-outage.html
[10] B. L Golden, E. A Wasil, and P. T. Harker, *The Analytic Hierarchy Process: Applications and Studies*, Berlin, Germany: Springer, 1989.
[11] H. S. Gunawi, T. Do, J. M. Hellerstein, I. Stoica, D. Borthakur, and J. Robbins, "Failure as a service (FaaS): A cloud service for large-scale, online failure drills," Univ. California, Berkeley, Berkeley, CA, Rep. no. UCB/EECS-2011–87, 2011.
[12] Y. Han and A. T. Chronopoulos, "A hierarchical distributed loop self-scheduling scheme for cloud systems," in *Proc. 12th IEEE Int. Symp. Netw. Comput. Appl.*, 2013, pp. 7–10.
[13] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, and J. Lawall, "Entropy: A consolidation manager for clusters," in *Proc. ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environments*, 2009, pp. 41–50.
[14] M. R. Hines and K. Gopalan, "Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning," in *Proc. ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environments*, 2009, pp. 51–60.
[15] OpenStack. (2018, Apr.). [Online]. Available: https://docs.openstack.org/
[16] Nova. (2018, Jun.). [Online]. Available: https://github.com/openstack/nova
[17] GoogleCloud. (2016, Jun.). [Online]. Available: https://www.status.cloud.google.com/incident/
[18] NRDC. (2014, Apr.). [Online]. Available: http://www.nrdc.org/energy/data-centerefficiency assessment.asp
[19] Jbb2005. (2014, May). [Online]. Available: http://www.spec.org/jbb2005/
[20] Web2005. (2014, May). [Online]. Available: http://www.spec.org/web2005
[21] TPCC. (2016, Jun.). [Online]. Available: http://www.tpc.org/tpcc
[22] J. Hu, J. Gu, G. Sun, and T. Zhao, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment," in *Proc. 3rd Int. Symp. Parallel Archit. Algorithms Program.*, 2010, pp. 89–96.
[23] K. Z. Ibrahim, S. Hofmeyr, C. Iancu, and E. Roman, "Optimized pre-copy live migration for memory intensive applications," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2011, Art. no. 40.
[24] K. Jackson, *OpenStack Cloud Computing Cookbook*. Birmingham, U.K.: Packt Publishing Ltd, 2012.
[25] X. Jiang and D. Xu, "VIOLIN: Virtual internetworking on overlay infrastructure," in *Proc. Int. Symp. Parallel Distrib. Process. Appl.*, 2005, pp. 937–946.
[26] A. Kangarlou, D. Xu, P. Ruth, and P. Eugster, "Taking snapshots of virtual networked environments," in *Proc. 2nd Int. Workshop Virtualization Technol. Distrib. Comput.*, 2007, pp. 1–8.
[27] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kVM: The linux virtual machine monitor," in *Proc. Linux Symp.*, 2007, pp. 225–230.
[28] K. Li, C. Liu, K. Li, and A. Y. Zomaya, "A framework of price bidding configurations for resource usage in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 8, pp. 2168–2181, Aug. 2016.
[29] C. Liu, K. Li, and K. Li, "Minimal cost server configuration for meeting time-varying resource demands in cloud centers," *IEEE Trans. Parallel Distrib. Syst.*, 2018.
[30] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "Live migration of virtual machine based on full system trace and replay," in *Proc. 18th ACM Int. Symp. High Perform. Distrib. Comput.*, 2009, pp. 101–110.
[31] H. Liu, H. Jin, C.-Z. Xu, and X. Liao, "Performance and energy modeling for live migration of virtual machines," *Cluster Comput.*, vol. 16, no. 2, pp. 249–264, 2013.
[32] A. B. Nagarajan, F. Mueller, C. Engelmann, and S. L. Scott, "Proactive fault tolerance for HPC with xen virtualization," in *Proc. 21st Annu. Int. Conf. Supercomput.*, 2007, pp. 23–32.
[33] M. Nelson, B.-H. Lim, G. Hutchins, et al., "Fast transparent migration for virtual machines," in *Proc. USENIX Annu. Tech. Conf., General Track*, 2005, pp. 391–394.
[34] S. Penmatsa, A. T. Chronopoulos, N. T. Karonis, and B. R. Toonen, "Implementation of distributed loop scheduling schemes on the teragrid," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2007, pp. 1–8.
[35] T. L. Saaty, "Decision making with the analytic hierarchy process," *Int. J. Serv. Sci.*, vol. 1, no. 1, pp. 83–98, 2008.
[36] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *IEEE Trans. Serv. Comput.*, vol. 3, no. 4, pp. 266–278, Oct.–Dec. 2010.
[37] M. M. Theimer, K. A. Lantz, and D. R. Cheriton, "Preemptable remote execution facilities for the v-system," *ACM*, vol. 19, no. 5, 1985.
[38] C. Weng, M. Li, Z. Wang, and X. Lu, "Automatic performance tuning for the virtualized cluster system," in *Proc. 29th IEEE Int. Conf. Distrib. Comput. Syst.*, 2009, pp. 183–190.
[39] T. Wood, P. J. Shenoy, A. Venkataramani, M. S. Yousif, et al., "Black-box and gray-box strategies for virtual machine migration," in *Proc. 4th USENIX Conf. Netw. Syst. Des. Implementation*, 2007, p. 17.



Ronghui Cao is currently working toward the PhD degree in the College of Computer Science and Electronic Engineering, Hunan University, China. His current research interests focus on cloud computing and virtualization technology.

Zhuo Tang received the PhD degree in computer science from the Huazhong University of Science and Technology, China, in 2008. He is currently an associate professor of Computer Science and Technology, Hunan University. His research interests include security model, parallel algorithms, and resources scheduling for distributed computing systems, grid and cloud computing. He is a member of the ACM, IEEE and CCF.

Kenli Li received the PhD degree in computer science from the Huazhong University of Science and Technology, China, in 2003. Now he is a professor of Computer Science and Technology, Hunan University, associate director of National Supercomputing Center in Changsha. His major research includes parallel computing, grid and cloud computing, and DNA computer. He is an outstanding member of CCF and a senior member of the IEEE.

Keqin Li is a SUNY distinguished professor of computer science. His current research interests include parallel computing and high-performance computing, distributed computing, cloud computing, and big data computing, etc. He has more then 520 research publications. He is currently or has served on the editorial boards of the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Computers*, the *IEEE Transactions on Cloud Computing*, the *IEEE Transactions on Services Computing*, the *IEEE Transactions on Sustainable Computing*. He is an IEEE fellow.