Jestr

Journal of Engineering Science and Technology Review 10 (5) (2017) 41 - 49

Research Article

JOURNAL OF
**Engineering Science and Technology Review**

www.jestr.org

# Study on Fault Tolerance Method in Cloud Platform based on Workload Consolidation Model of Virtual Machine

**Zhixin Li[1,2,*], Lei Liu[1] and Zeyu Tong[3]**

[1]*College of Computer Science and Technology, Jilin University, Changchun 130012, China*
[2]*School of Computer Technology and Engineering,Changchun Institute Of Technology, Changchun 130012, China*
[3]*Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, 21218, United States*

Received 21 April 2017; Accepted 7 October 2017

***Abstract***

The fault tolerance method of virtual machines (VM) guarantees reliability to the service capability of cloud platforms. VM workloads are dynamic and uncertain, and thus, they affect the reliability and task processing capability of entire cloud platforms. In this study, a fault tolerance method based on the VM workload consolidation model was proposed to solve problems concerning the reliability and task processing capability of cloud platforms caused by VM workloads, thus improving the reliability of VMs and overall performance of cloud platforms. First, the method was analyzed on the basis of the distinct relationship of VM workload and VM reliability and task processing capability. Then, the workload state of VM was predicted and analyzed by linear regression using VM workload monitoring data, and the VM workload consolidation algorithm was constructed based on expected workload constraint and optimization of fault tolerance time. Finally, the fault tolerance method based on the VM workload consolidation model was compared with the Radom method and the Max method. Research results demonstrate the potential of the proposed method to improve VM reliability in cloud platforms by 20% and 47% compared with those for the Radom and Max methods, respectively. In the same workload phase, the task completion rate of the proposed method increased significantly (15% and 30%, and 22% and 30%), and the percentages were higher than those for the Radom and Max methods, respectively. Moreover, the proposed method shortened task response time. This study concludes that the workload consolidation of VMs can increase the reliability and task processing capability of VMs. This proposed method can provide technological support to the fault tolerance of VMs in cloud platforms.

*Keywords:* cloud computing, virtual machine, workload consolidation, fault tolerance

## 1. Introduction

With the rapid development of cloud computing technology, an increasing number of enterprises has started to offer VM services in cloud platforms [1]. The reliability of VMs could directly influence cloud platform application services, such as e-mails, databases, web applications, and so on [2]. Subsequently, the fault tolerance method of VMs can effectively prevent failures in cloud platforms. Fault tolerance can also prevent the proliferation of error logs from one VM system to the next; consequently, cloud-based services that are continuously offered to the outside world are also improved, thus increasing the reliability of the entire cloud platform. However, combining the fault tolerance of VM workloads is necessitated prior the deployment of VM fault tolerance in cloud platforms. The workload state of VMs in cloud platforms is dynamic and uncertain [3,4]. Therefore, understanding how fault tolerance is deployed on the basis of the combined relationships of VM workload and VM reliability and task processing capability is imperative.

The fault tolerance method of VMs in cloud platforms was developed under the abovementioned circumstance. At present, the existing fault tolerance method selects possibilities of VM errors according to the physical state of a server where a VM is located. However, even if it attempts to solve VM reliability problems, the existing method often fails to establish the optimization of VM fault tolerance time and effective consolidation of VM workload state on fault tolerance. The optimization of VM fault tolerance time was studied by using tolerance methods and models [5] to tolerate faults upon the assumption of possible errors. However, without the allocation and consolidation of VM workloads, the approach frequently caused fault tolerance of VMs, which then affected the system performance of cloud platforms. In discussing the effects of VM workload on fault tolerance [6], the VM workload layout was optimized by measuring network transmission speed and delay, and consequently, to save on bandwidth and increase efficiency. However, the layout neglected the impacts of CPU and memory workload, as well as fault tolerance time on fault tolerance, with the consolidation of VM workloads.

In this study, a fault tolerance method is established according to the relationships of VM workload and VM reliability and task processing capability. The proposed method not only counterbalances the impacts of VM workload on fault tolerance to some extent, it also solves issues concerning the optimization of fault tolerance time. The proposed method also improves the reliability of VMs

*E-mail address: lizx12@mails.jlu.edu.cn
ISSN: 1791-2377 
doi:10.25103/jestr.105.05

in cloud platforms and the overall system performance of cloud platforms.

## 2. State of the art

Past works have reported a number of reliability issues of cloud platforms caused by VM workload [7], as well as the importance of VM fault tolerance to safeguard the reliability and available capability of cloud platforms (e.g., EC2 [8], Google App Engine [9], VMware [10], Xen [11], KVM [12], etc.). Meanwhile, most existing research has focused on the reactive and preventive fault tolerance methods [13]. In the reactive fault tolerance method, backing up the fault tolerance is considered a common approach. With respect to VM backups, Xu et al. [14] proposed the strategic planning of models and adaptive backups based on a genetic algorithm to shorten backup time. Cully et al. [15] used backups for entire cluster states by considering a workload phase, and then realized fault tolerance by recovering the latest check point for a cluster after error identification. Machida [16] reduced the number of backup servers by optimizing the layout of redundant VMs to ensure reliability (i.e., master and vice nodes were backed up in intervals for fault tolerance). However, the method was costly and necessitated an optimization of fault tolerance time. In the preventive fault tolerance method, fault tolerance is conducted following a resource workload state analysis [17]. VMs are transferred to another server to ensure normal service operations, and this is carried out by studying fault tolerance trigger points and the fault tolerance model. The preventive fault tolerance method also mainly involves workload balancing technology [18, 19], energy saving technology [20], system consistence technology [21], and so on. Zhang et al. [22] analyzed a running resource state with the hidden Markov model, calculated the future running state probability of a system, and conducted dynamic adjustments in system resource allocation to increase the efficiency of VM recovery against failures. Mallick et al. [23] performed clustering analysis on resource states using different numerical ranges of historical resource workload indices. A cluster was designated to a state point, and the short-term resource state was predicted with the Markov model. Bruneo et al. [24] reported that VM could be restarted regularly with a VM software recovery strategy to solve the aging failure of VM caused by workload and protect the availability of VM. Wu et al. [25] proposed a developmental method for model-based fault tolerance and realized seven middle fault-tolerance-mechanisms in cloud platforms, thus realizing the cross-platform characteristics of the fault tolerance mechanism. However, those methods could neither protect the fault tolerance time of VMs nor guarantee effective workload consolidation, and they were also costly for system operations. In addition, system reliability decreased when the workload pressure of the cloud platform system was large.

To solve the above reliability problems, this study proposes the reduction of VM workload by using a VM workload consolidation algorithm that is based on the relationship of VM workload and VM reliability and task processing capability. When the server workload expectation exceeds the threshold, the VM system adopts fault tolerance processes in anticipation of VM errors. The fault tolerance time of VM is optimized to reduce fault tolerance frequency, thereby protecting the availability and improving the performance of the cloud platform system.

This study is presented as follows: Section 3 introduces the measurement of the server-VM workload, the VM workload prediction model, the reliability model, and the fault tolerance method. Section 4 presents the experiment and the result analysis. Section 5 provides the conclusions.

## 3. Methodology

### 3.1 Measurement of server-VM workload

VM workload is strongly dynamic, and thus, measuring resource workload can effectively identify the state of resources. Establishing whether or not a VM is overloaded can be determined by the resource states. Therefore, choosing the appropriate measurement for resource workload can enhance workload consolidation and increase VM reliability.

Definition 1: Workload phase. Workload phase refers to a time interval during VM operation. The VM workload is relatively stable in a single workload phase, and this is expressed as $l_j=\{l_1,...l_j,l_{j+1},...\}$.

This definition can be used to measure the change rate of VM workload in the workload phase. When the VM is in running state, the change in workload from phase $l_{j-1}$ to phase $l_j$ reflects the workload state in this particular workload phase.

Definition 2: Server workload expectation. Workload expectation refers to the average measure of server computing resources that are occupied by the VM after virtualization.

In cloud platform systems, the resource pool is composed of a series of servers. Each server is expressed by $S_i$ and the server resource cluster of cloud environment is $S=\{S_1,S_2...S_n\}$. In the initial state, each node server $S_i$ is assumed to be allocated with $k$ VMs in $S_i(vm_1...vm_k)$, where $vm_k$ represents the number of VMs. Each server $S_i$ contains multiple hardware resources, such as CPU, memory, network, and so on.

The VM workload discussed in this study mainly refers to two hardware resources: CPU and memory. The proportion of CPU and memory of VM are expressed as $C_{vcpu}$ and $C_{vmem}$. The CPU and memory resources of each VM are regularized to 1.

VM workload is calculated based on CPU utilization $\lambda_{vcpu}$ and memory utilization $\lambda_{vmen}$. The workload of $vm_k$ on node server $S_i$ in the workload phase $l_j$ is expressed as:

$$W_{i,k}(l_j)=\lambda_{vcpu}C_{vcpu}+\lambda_{vmen}C_{vmen}. \quad (1)$$

The calculated workload of server $S_i$ in workload phase $l_j$ is the workload sum for all VMs as follows:

$$W_i(l_j)=\sum_{k=1}^{m}W_{i,k}(l_j). \quad (2)$$

In this study, the server workload expectation is defined as:

$$S(l_j)=E(\frac{W_i(l_j)}{m}), \quad (3)$$

where m is the number of VMs on server $S_i$.

Server workload expectation reflects the average proportion of VM workload on server resources. When $S(l_j)$ is exceedingly large, the VMs occupy relatively more server resources in workload phase $l_j$ and thus are overloaded. An overload constraint ($\Omega$) is set for the overload decision strategy $S(l_j) < \Omega$, where $\Omega$ denotes the degree of overload. Given that the total workload of the server in workload phase $l_j$ is lower than $\Omega$, i.e., $0< \Omega <1$, the VM resource allocation attempts to reduce the degree of overload. In addition, given that the expression claims extensive system resource consumption for the migration fault tolerance of VMs, the system can tolerate a certain degree of overload. However, system performance declines significantly when overloading is reached at a certain point, and this occurrence reduces reliability.

### 3.2 VM workload prediction model

The VM workload is strongly correlated with time on the basis of the VM workload measurement; in other words, the VM workload of the previous workload phase significantly affects those of the next phase. Therefore, VM workload can be predicted by linear regression. The timeliness and accuracy of VM workload prediction for the succeeding workload phases not only affect the cloud computation of VM reliability, they also influence the optimization of VM fault tolerance time.

In this study, $W_{i,k}(l_1)$ denotes the initial workload of $vm_k$ on server $S_i$, while $W_{i,k}(l_j)$ denotes the VM workload in workload phase $l_j$. Based on the changes in the VM workload for a workload phase:

$$W_{i,k}(l_j)=W_{i,k}(l_{j-1})+\varpi(l_j), \tag{4}$$

where $\varpi_{i,k}(l_j)$ is the observed change rate of workload of $vm_k$ on server $S_i$ in workload phase $l_j$. In previous workload phases $l_j=\{l_1,...l_j,l_{j+1},...\}$, the workload variation set is $\varpi_{i,k}(l_j)=\{\varpi_{i,k}(l_2),\varpi_{i,k}(l_3),...\varpi_{i,k}(l_j)\}$.

VM workload is predicted by linear regression on the basis of VM dynamics. From the definition of general linear regression, the estimation function is:

$$\hat{\varpi}_{i,k}(l_{j+1}) = \theta_0 + \theta_1 * l_{j+1}, \tag{5}$$

where $\hat{\varpi}_{i,k}(l_{j+1})$ is the predicted VM workload change rate of the next workload phase. From the VM workload change rates in previous phases, the coefficients $\theta_0$ and $\theta_1$ are obtained by solving the linear regression equation with the least square method. Moreover, $\theta_0$ and $\theta_1$ change with historical workload change rates. By using this method, dynamically adjusting the parameters of the linear regression model based on workloads in the latest phase is feasible. Thus, the method can adapt to workload fluctuations. we define $\theta_0$ and $\theta_1$ in the following way:

$$\theta_0 = \frac{\sum l_j^2 \sum \varpi_{i,k}(l_j) - \sum l_j \sum l_j * \varpi_{i,k}(l_j)}{n\sum l_j^2 - (\sum l_j)^2} \tag{6}$$

$$\theta_1 = \frac{n\sum l_j * \varpi_{i,k}(l_j) - \sum l_j \sum \varpi_{i,k}(l_j)}{n\sum l_j^2 - (\sum l_j)^2}. \tag{7}$$

On the basis of workload change rate $\hat{\varpi}_{i,k}(l_{j+1})$ in the next workload phase $l_j$, the predicted VM workload $\hat{W}_{i,k}(l_{j+1})$ can be calculated from equation (4). The server workload expectation $\hat{S}(l_{j+1})$ can be obtained from equation (3). If $\hat{S}(l_{j+1}) < \Omega$, then the server is not overloaded in workload phase $l_{j+1}$; that is, the server is running reliably, and workload consolidation allocation is conducted.

### 3.3 VM reliability model

Given that server reliability is a random distribution event of time, the faults are influenced by sudden workload increase in cloud platform systems and errors accumulate of VMs. These faults are considered as random faults, such as system breakdown caused by sudden increases in VM workload, which occur occasionally. Studying the system fault log [26-28] showed that server reliability complied with Weibull distribution. In this study, server reliability distribution was verified by an experiment based on the reliability analysis of VMs in cloud platforms.

Definition 3: VM reliability refers to the probability of VM resources to accomplish assigned functions, and it is one of the main evaluation indices of reliability of VM resources in running state. In this study, the absence of a VM error log is considered in VM reliability measurement; that is, $R_{vm_k}$ reflects the reliability of $vm_k$.

The Weibull probability density distribution function of the two parameters is:

$$f(l_j) = \frac{m}{\eta}(\frac{l_j}{\eta})^{m-1} e^{-(\frac{l_j}{\eta})^m} \quad m,\eta>0. \tag{8}$$

The reliability function is:

$$R_{vm_k}(l_j) = e^{-(\frac{l_j}{\eta})^m}, \tag{9}$$

where $m$ is the shape parameter; $\eta$ is the scale parameter; and $l_j$ is the workload phase. VM reliability with server workload expectation can be expressed as:

$$R_{vm_k}(l_j) = R\{t \le l_j \mid S(l_j)\}. \tag{10}$$

The VM reliability in the model is designated with a lower value than the system threshold for fault tolerance. Let:

$$R_{vm_k}(l_j) < \Phi, \tag{11}$$

where $\Phi$ is the threshold of system reliability.

When the reliability value of $vm_k$ is smaller than the threshold of system default, fault tolerance occurs. Choosing the appropriate fault tolerance time increases system reliability and reduces system performance loss.

### 3.4 Fault tolerance method

The future running state of VMs is predicted by using established models on VM workload and reliability in cloud platforms. In this section, the VM fault tolerance method is described. The system structure of the VM fault tolerance model is shown in Fig. 1. The Workload Manager module is implemented for workload management and consolidation, while the Failure Detector module is used for error detection.

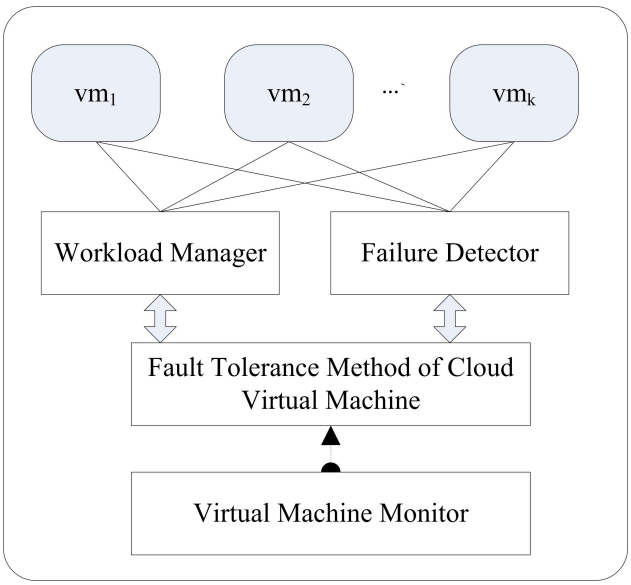


**Fig. 1.** System structure of the VM fault tolerance model

#### 3.4.1 Basic principle behind the fault tolerance method

The main principle behind the fault tolerance method is related to the analysis of relationships of VM workload and VM reliability and task processing capability. By using a mathematical model, the mutual relationships among the above parameters are verified.

Definition 4: Task processing capability of VM ($\mu$). A high $\mu$ of VM resources in unit time reflects strong resource service capability. In the expression, $W_{i,k}(l_j)$ is the VM workload; $R_{vm_k}$ is the reliability of $vm_k$; and $h_{vm_k}$ is the task completion rate of $vm_k$. $\mu$ is then defined as:

$$\mu_{i,k} = R_{vm_k}(l_j) * \ln(1/h_{vm_k}) + W_{i,k}(l_j). \quad (12)$$

The task processing capability of VMs reflects the relationship between resource workload and reliability.

Theorem 1: If two VMs have the same $\mu_{i,k}$ and $W_{i,k}(l_j)$, where $R_{vm_k}$ and $R^*_{vm_k}$ denote high and low reliabilities of VM ($R_{vm_k} > R^*_{vm_k}$) while $h_{vm_k}$ and $h^*_{vm_k}$ denote the task completion rates of VM under high and low reliabilities, then $h_{vm_k} > h^*_{vm_k}$.

Proof: According to known conditions, if two VMs have the same $\mu_{i,k}$ and $W_{i,k}(l_j)$, then the following can be derived from equation (12): $h_{vm_k} = e^{-(\mu_{i,k} - W_{i,k}(l_j))/R_{vm_k}}$ and $h^*_{vm_k} = e^{-(\mu_{i,k} - W_{i,k}(l_j))/R^*_{vm_k}}$.

Given that $R_{vm_k} > R^*_{vm_k}$, we can deduce that $h_{vm_k} > h^*_{vm_k}$.

Theorem 1 indicates that the higher the VM reliability is, the higher the task completion rate will be. In other words, increasing VM reliability can protect the task completion rate of the cloud platform.

Theorem 2: Given the same context and constant task processing capability $\mu_{i,k}$ and task completion rate $h_{vm_k}$ of VMs in a single workload phase, if $W_{i,k}(l_j) < W_{i,k}(l_{j+1})$, then $R_{vm_k}(l_j) > R_{vm_k}(l_{j+1})$.

Proof: According to known conditions (i.e., the same context and constant task processing capability $\mu_{i,k}$ and task completion rate $h_{vm_k}$ of VMs in a single workload phase), equation (12) can be used to derive:

$$R_{vm_k}(l_j) = (\mu_{i,k} - W_{i,k}(l_j)) / \ln(1/h_{vm_k}) \text{ and}$$
$$R_{vm_k}(l_{j+1}) = (\mu_{i,k} - W_{i,k}(l_{j+1})) / \ln(1/h_{vm_k}).$$

Given that $W_{i,k}(l_j) < W_{i,k}(l_{j+1})$, we can deduce that $R_{vm_k}(l_j) > R_{vm_k}(l_{j+1})$.

Theorem 2 suggests that the VM workload in the workload phase during the running state of VM resources is negatively correlated with VM reliability. Therefore, workload consolidation and allocation are needed to reduce VM workload and improve VM reliability.

Theorems 1 and 2 establish the relationship of reliability, workload, and task completion rate of VMs. Subsequently, workload consolidation and allocation are introduced.

Theorem 3: Given a task resource request $\{\tau_q, q = 1...r\}$, if this request is accomplished by $n_1$ physical machines ($S=\{S_1, S_2 ... S_{n_1}\}$) or by the same quantity of VMs ($SP_i=\{vm_1 ... vm_{n_1}\}$), where the number of servers required for the VM to settle is $n_2$ ($n_2 \le n_1$), then the reliability of the VM ($R_{vm}$) is smaller or equal to reliability of server ($R_S$), i.e., $R_{vm} \le R_S$.

Proof: From section 3.3, if the fault probability of a service to $n_1$ physical machines obeys Weibull distribution $F - (n_1, R_s)$, then the expectation and variance of the fault probability of physical machines in the workload phase are $E(F) = (n_1, \eta\Gamma(1 + \frac{1}{m}))$ and $Var(F) = (n_1, \eta^2[\Gamma(\frac{1}{m} + 1) - \Gamma^2(\frac{1}{m} + 1)])$.

If $n_2$ physical machines ($n_1 \ge n_2$) are used, and each physical machine involves $m$ VMs, ($n_1 = n_2 * m$), then the VMs on these physical machines cannot run normally (i.e., faults are observed on the physical machines). Thus, the fault probability of VMs obeys Weibull distribution $F_1 - (n_1, R_{sp}(t))$, which is the same as the physical distribution. After virtualization, the mathematical expectation of fault probability in the workload phase is $E(F_1) = E(n_2, \eta\Gamma(1 + \frac{1}{m}))$. The corresponding variance is:

$$Var(F_1) = \frac{n_1}{m} Var(Y_{1...m} + Y_{m+1...2m} + Y_{n_1-m+1...n_1})$$
$$= (n_1 m, \eta^2[\Gamma(\frac{1}{m} + 1) - \Gamma^2(\frac{1}{m} + 1)]).$$

Expectedly, the following can be obtained: $Var(F) \ge Var(F_1)$, $R_{vm} \le R_S$.

Results indicate that the higher the number of VMs on each server is, the higher the variance will be. The variances of the VM and server are equal only when there is one VM

on one server. By referring to the variance, the reliability of the VM is lower than that of the physical machine. When multiple VMs are operated on a physical server, the hardware fault affects more applications compared with cases wherein each physical server is responsible for a single task only. Hence, in the cloud platform environment, the reliability of the VM is lower than that of the server. In other words, the likelihood to develop the functions of a service through VM is lesser compared with that through the server.

Theorem 4: Given task resource request $\{\tau_q, q=1...r\}$, if the task processing capability of a VM is $U_{i,k}$ in $U_{i,k}=\mu_{i,k}+\sum_{q=1,,q\neq k}^{r}\tau_q$, then the average workload phase for task processing is $l=\sum_{q=1,,q\neq k}^{r}\tau_q/U_{i,k}$. If $\tau_q$ is distributed on $k$ VMs uniformly, then the task processing capability is $\{\mu_{i,1}...\mu_{i,k}\}$, where $\mu_{i,k}$ is the task processing capability of $vm_k$ on server $S_i$. The average workload phase for the task processing is $\hat{l}=\sum_{q=1}^{r}\tau_q/n_1\mu_{i,k}$, where $n_1$ is the total number of VMs. Then, $U_{i,k}\geq\max_{k=1}^{k}\mu_{i,k}$ and $l\geq\hat{l}$.

Proof: Suppose the current VM is $vm=vm_{i,1}$ and its task processing capability is $U_{i,1}$, then $U_{i,1}=\mu_{i,1}+\sum_{q=1,,q\neq 1}^{r}\tau_q$.

When task request $\tau_q$ is considered for a VM, the workload of this VM increases. The task processing capability is $U_{i,1}=R_{vm_1}(l_j)*\ln(1/h_{vm_1})+W_{i,1}(l_j)$.

A specific VM workload is higher than the average VM workload, which is allocated by tasks. On the basis of equation (3), the workload of the specific VM is higher or equal to the average server workload expectation $W_{i,1}(l_j)\geq S(l_j)$. To meet the task demand, $U_{i,1}\geq\mu_{i,1}$. Given that VMs have the same configuration, then $U_{i,1}\geq\{\mu_{i,1},...\mu_{i,k}\}$; that is, $U_{i,k}\geq\max_{k=1}^{k}\mu_{i,k}$.

Allocating tasks to a few VMs enhances VM task processing capability. However, processing time also increases due to limited resources. Therefore, $l\geq\hat{l}$.

Theorem 4 shows that the average distribution of tasks on different VMs can shorten task processing time and increase processing efficiency. Whether or not the task is allocated to VMs on a single server or to VMs on different servers is determined by Theorem 5.

Definition 5: The average task processing capability $P(sp_i)$ of server $S_i$ in $sp_i=\{vm_1...vm_{k_1}\}$ is the sum of VMs on server $S_i$:

$$\begin{aligned}P(sp_i)&=\sum_{k_1=1}^{k_1}\mu_{i,k_1}/k_1\\&=\sum_{k_1=1}^{k_1}(R_{vm_{k_1}}*\ln(1/h_{vm_{k_1}})+W_{i,k_1}(l))/k_1\ .\end{aligned}\tag{13}$$

Theorem 5: Given a task resource request $\{\tau_q, q=1...r\}$ in which $n_1$ and $n_2$ servers are used, then $n_1>n_2$. The same quantity of VMs ( $sp_{\{i,i=1..n_1\}}=\{vm_1...vm_{k_1}\}$ and $sp_{\{i,i=1..n_2\}}=\{vm_1...vm_{k_2}\}$ ) are deployed on the server. The average task processing capability is $P(sp_{n_1})$ when a task is uniformly allocated to VMs of different servers, and it is $P(sp_{n_2})$ when a task is allocated to VMs of only a few servers. To meet task completion rate $h_{vm_k}$ and server local expectation $\Omega$, we derive $P(sp_{n_1})>P(sp_{n_2})$.

Proof: Given that $n_1>n_2$ and the total number of VMs is $k$, the average number of VMs on each server is $(sp_{n_1}=k_1=k/n_1)<(sp_{n_2}=k_2=k/n_2)$. the average task processing capability is:

$$\begin{aligned}P(sp_i)&=\sum_{k_1=1}^{k_1}\mu_{i,k_1}/k_1\\&=\sum_{k_1=1}^{k_1}(R_{vm_{k_1}}*\ln(1/h_{vm_{k_1}})+W_{i,k_1}(l))/k_1\\&=\sum_{k_1=1}^{k_1}(R_{vm_{k_1}}*\ln(1/h_{vm_{k_1}})/k_1+S_i(l)\ .\end{aligned}$$

According to Theorem 3, the reliability of using VMs on many servers is higher than that of using VMs on a few servers only; that is, $R_{vm_{k_1}}>R_{vm_{k_2}}$. On the basis of equation (3), the server workload expectation meets $S_i(l)<\Omega$.

Therefore, $P(sp_{n_1})>P(sp_{n_2})$ is proven.

Theorems 3, 4 and 5 prove that allocating a task to different VMs on servers during workload consolidation and task allocation not only increases the reliability of VMs, it also shortens task processing time and improves processing efficiency.

### 3.4.2 Optimization of fault tolerance time

Fault tolerance of VMs results in system performance loss. Therefore, the fault tolerance time necessitates optimization to reduce fault tolerance frequency, thereby relieving influences on system performance and protecting normal service operations.

In this study, $l'$ represents fault tolerance time, which is selected from the changes in predicted workloads of the prediction model mentioned in Section 3.2. If the predicted server workload expectation $\hat{S}(l')$ is higher than the workload expectation of system default $(\Omega)$, i.e., $\hat{S}(l')>\Omega$, then the fault tolerance of VM is implemented in $l'$. The best fault tolerance time in this study is determined by Theorem 6.

Theorem 6: If the initial workload of $vm_k$ on server $S_i$ is $W_{i,k}(l_1)$ and the average change rate of $vm_k$ in previous workload phases $l_j$ is $\bar{w}$, then the fault tolerance time of $vm_k$ is:

$$l'=(\Omega-W_{i,k}(l_1)-(j-1)*\bar{w}-\theta_0)/\theta_1,\tag{14}$$

where $\Omega$ is the upper limit of workload expectation; $W_{i,k}(l_1)$ is the initial workload; and $\bar{w}$ is the average workload change rate. Coefficients $\theta_0$ and $\theta_1$ are calculated from equations (6) and (7).

The optimal workload phase for VM fault tolerance time in equation (14) is influenced by two factors: time and spatial changes. This approach implies that VM fault tolerance time is related to workload changes and VM workload on a few servers. To reduce fault tolerance

frequency, the lowest reliability of VM is chosen for fault tolerance.

### 3.4.3 Fault tolerance method based on workload consolidation

The task resource request $\{\tau_q, q = 1...r\}$ is managed by the Workload Manager module. According to Theorem 5, a task is allocated to different VMs on different servers by the Workload_Consolidaton function to reduce VM workload and increase reliability. The optimization of fault tolerance time ($l'$) is determined by the VM workload prediction model. Thus, the proposed method not only employed fault tolerance to VMs with errors, it also effectively reduced fault tolerance frequency and system cost, as well as improved system performance. The script of the fault tolerance method is shown in Fig. 2.

```
void Workload_Consolidation()
{
If (τ_q <= 0 )  //The number of tasks must be greater than 0
{
cout << "The initial value is incorrect."<< endl;
return;
}
      int serverNO =n;//There are n servers. S={S1,S2...Sn}
      int vmNO = k;//Each server has k virtual machines. Si(vm1...vmk)
      int v1 = 0;//Representing the assigned server number.
      int v2 = 0;//Representing the assigned virtual machine number.
      If (taskNO > serverNO*vmNO)//The number of tasks is greater than the
                                   total number of virtual machines.
{
      for (int i = 1; i <= serverNO*vmNO; i++)

      v1 = i%serverNO == 0 ? serverNO:i%serverNO;
                              //Calculating the server number to which a task
                                is assigned.
      V2 = (i - 1) / serverNO + 1;//Calculating the virtual machine number.
      }}
      int main()
      {
      workload_Consolidation(τ_q);

      Calculating  workload expectation  𝕊(l_j)   based on the formula (3)

      If( 𝕊(l_j) >  Ω   ) then

        Selecting VM fault tolerance based on the formula (11)

        Selecting VM fault tolerance timing based on the theorem (14)

      return 0;
        If  (Request  donot  Fit any VM) then
        Add new VM
      }
```

**Fig. 2.** Fault tolerance method based on VM workload consolidation

## 4. Result analysis and discussion

The validity of the proposed method is verified from these three aspects: (1) evaluation and analysis of workload prediction, (2) analysis of VM reliability based on workload consolidation, and (3) comparison of different allocation strategies of task resource requests in the experiment to verify the validity of the proposed fault tolerance method. The three methods are compared in the following experiments:

(1) Random: a task resource request is randomly allocated to a VM;

(2) Max: concentrated allocation of a task resource request to further allocate the task of a few VMs to the maximum degree; and

(3) Workload_Consolidation (proposed method): the task is averaged and allocated to the different VMs of different servers in proper order.

### 4.1 Deployment of experimental environment

Two Daylight A840-G10 servers are used for the service platform (CPU: AMD 6376, 16 core 2.3 GHz × 4; memory: 256 G and Gigabit LAN; disk array: Daylight DS800-G25 and 41.6 TB storage capacities). All server nodes are connected by gigabit optical fiber exchangers.

The software environment is deployed by two servers using Xen virtualization platform. Forty VMs are initialized (Table 1):

The Red Linux 5.0 operation system and Nigix application service program are installed into the VMs, and a distributed website (JDK1.6 edition) is established.

The CPU-bound jobs are calculated through JMeter simulation. An analog pressure workload is launched to test the VM workload and collect data. The pressure workload, which is generated by the client end, allows the VMs with Nigix to operate with full workload.

**Table 1.** Deployment of software environment

| Xen | VM | CPU | Core | Menmory |
|---|---|---|---|---|
| 5.0 version | 40 | 1 | 2 | 2G |

### 4.2 Analysis of experimental results

First, the validity of the proposed basic workload evaluation strategy is evaluated. The mean square error (MSE) is used as an evaluation index of performance prediction:

$$MSE = \frac{\sum_{t=l_1}^{t=l_j} (y_t - \hat{y}_t)^2}{\sum_{t=l_1}^{t=l_j} (y_t)^2}. \quad (15)$$

where $\hat{y}_t$ is the predicted workload; $y_t$ is the actual workload; and $l_j$ is the workload phase (total number of predictions).

#### 4.2.1 Workload prediction analysis

In this experiment, 20 low-workload VMs and 20 high-workload VMs are used to evaluate workload prediction. The length of the workload phase is 250. The workload expectations of the low-workload server and the high-workload server are shown in Fig. 3 and Fig. 4, and the MSE values are 0.122 and 0.019, respectively. According to experimental results, the MSE of the prediction error is controlled in the acceptable range and reflects the validity of the established model. In addition, the MSE is increased gradually with the continuous growth of $l_j$, which indicates that appropriate workload phases increases prediction accuracy. When the workload is significantly changed, the prediction accuracy is lowered to some extent, but no influence is established for workload consolidation.

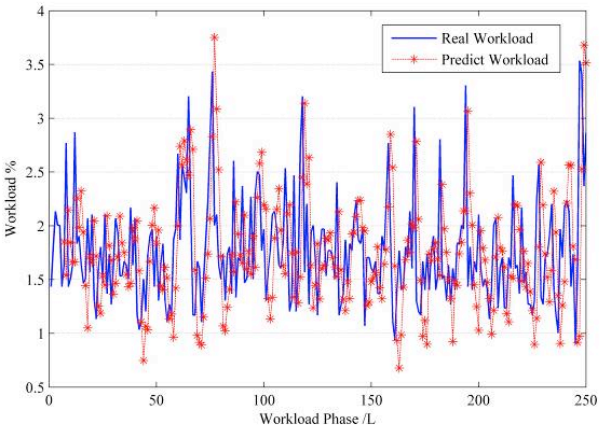


**Fig. 3.** Actual and predicted workload expectations of low-workload servers

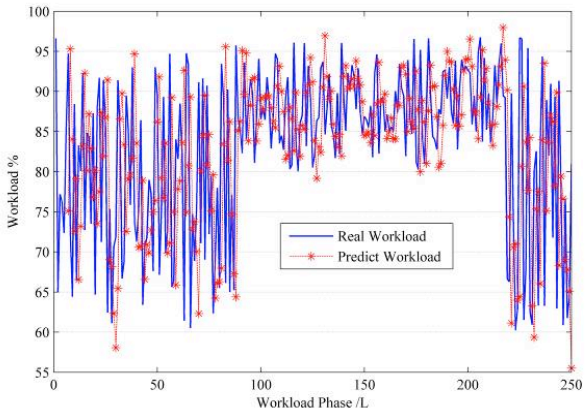


**Fig. 4.** Actual and predicted workload expectations of high-workload servers

#### 4.2.2 Reliability analysis of VMs in workload conditions

In this experiment, 20 low-workload VMs and 20 high-workload VMs are used to evaluate VM reliability. The experimental results are shown in Fig. 5. When the VM workload is lower than that of the low-workload VMs, no accumulation errors are established in terms of observation time. In addition, VM reliability is nearly 100%. In high-workload conditions, the accumulation probability of VM error logs in the entire cloud platform is lowest (approximately 38%) in terms of observation time. The experimental results reflect that the VM fault is related to the increase of cloud computing system workload.

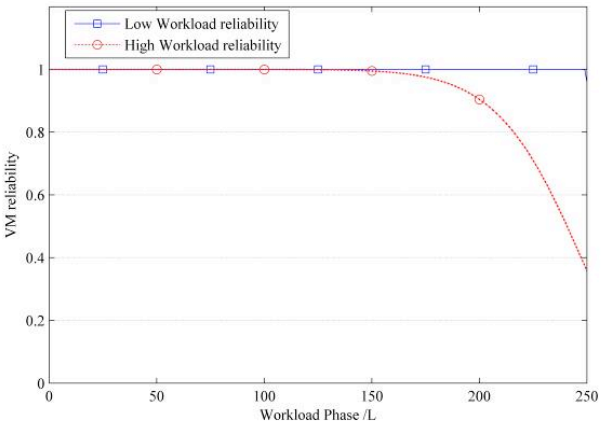


**Fig. 5.** Reliability of high-workload and low-workload VMs

#### 4.2.3 Reliability analysis of the fault tolerance method based on workload consolidation

The thresholds of server workloads are set to 0.9, 0.8 and 0.7. Then, the VM workloads are consolidated and the VM fault tolerance time is optimized, as depicted by Theorem 6. The experimental results are shown in Fig. 6. The reliability of VM is increased by workload consolidation. Accordingly, VM reliability is negatively correlated with the server workload threshold. Therefore, the fault tolerance method based on workload consolidation can increase VM reliability.

#### 4.2.4 Comparative analysis of fault tolerance method based on workload consolidation

The reliability of workload expectation threshold of different servers was previously verified. In this section, the Random, Max, and Workload_ Consolidation methods are compared in terms of VM reliability and VM task completion rate. The number of VMs, quantities of required jobs, and workload phases are shown in Table 2.

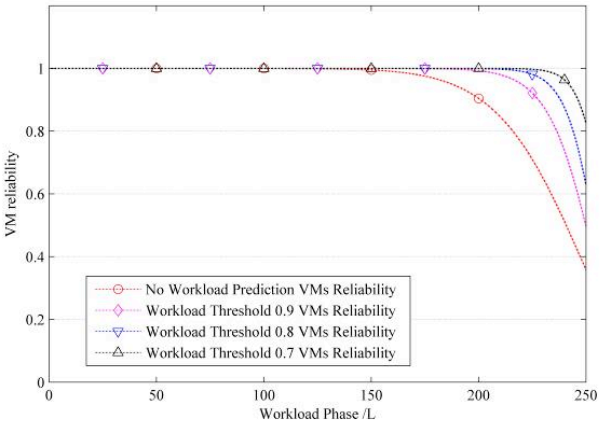


**Fig. 6.** Reliability of VMs based on workload consolidation

**Table 2.** Node specifications, requested jobs, workload phases

| $S_i(vm_1...vm_k)$ | $\{\tau_q, q=1...r\}$ | Workload phase ($l_j$) |
|---|---|---|
| $S_1$ = {20} | $r$ = {40,80,160,320} | {100,150,200,250} |
| $S_2$ = {20} | $r$ = {40,80,160,320} | {100,150,200,250} |

**Table 3.** Reliability of VMs in the three methods

| phase \ method | Workload Consolidation | Random | Max |
|---|---|---|---|
| Reliability(100) | 100% | 100% | 100% |
| Reliability(150) | 100% | 100% | 92.6% |
| Reliability(200) | 100% | 89.5% | 73.4% |
| Reliability(250) | 82.5% | 62.5% | 35.5% |

The reliability of VMs in workload phases is analyzed using the above three methods (Table 3). A relatively long running time of loaded VMs results in a relatively low reliability. At workload phase 250, the accumulation error probabilities of Workload_Consolidation, Random, and Max, are 82.5%, 62.5%, and 35.5%, respectively. Therefore, VM reliability is improved by the proposed workload consolidation algorithm, the rates of which are 20% and 47% higher than those by the Radom and Max methods, respectively.

The task completion rates of the three methods in the regulated workload phase are shown in Fig. 7. A comparison of task completion rates for different job quantities ($r$ = {40,80,160,320}) in workload phase $l_j$ =100 is shown in Fig. 7(a), and the task completion rates in workload phase $l_j$ ={150, 200, 250} are shown in Fig. 7(b), Fig. 7(c), and Fig. 7(d). As shown by Fig. 7, Workload_Consolidation obtained a higher task completion rate than the two other methods. At $l_j$ =250 and $r$ = {160,320}, the task completion rates of Workload_Consolidation is 15% and 30% higher than that of Radom and 22% and 30% higher than that of Max.

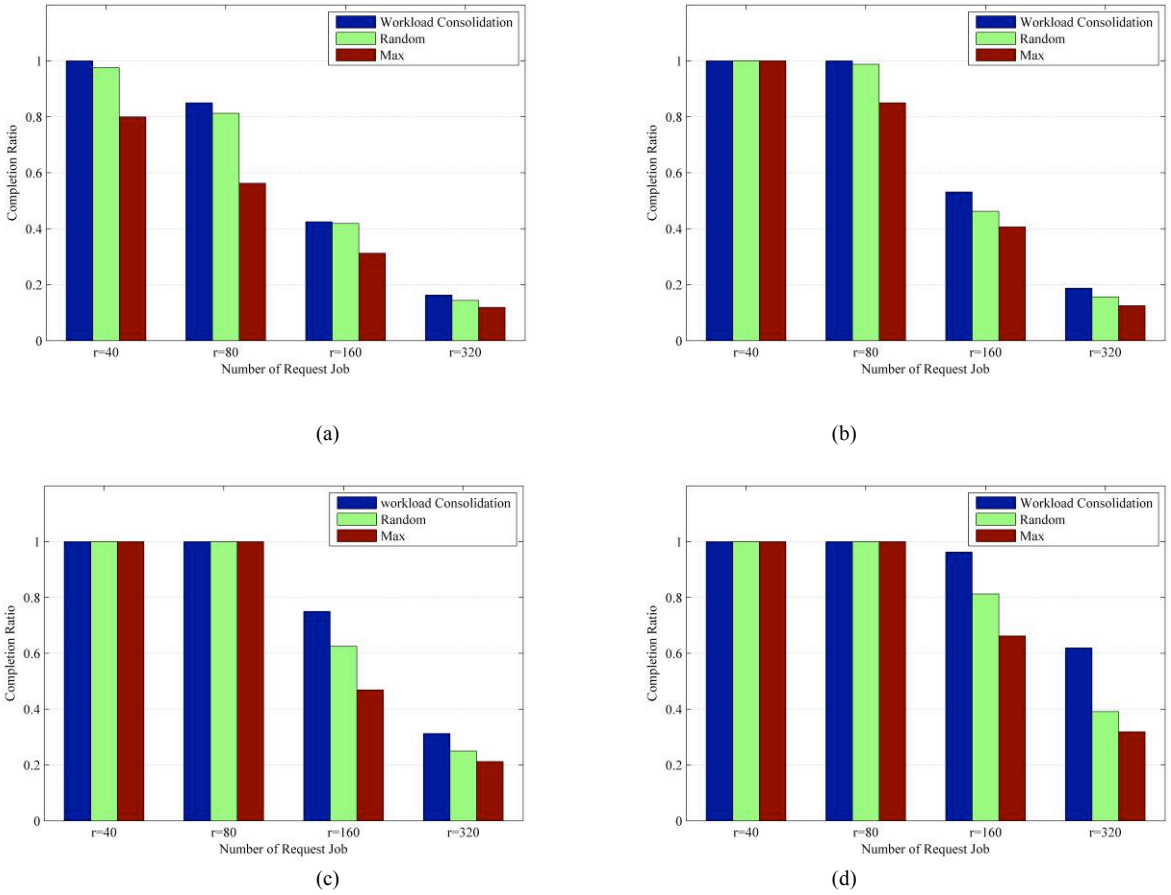


**Fig. 7.** Comparison of task completion rates

### 4.2.5 Validity of fault tolerance method

This experiment is conducted to verify the validity fault tolerance method based on the workload consolidation model, and ultimately, to improve service capability during system operations. In the experiment, JMeter simulates workloads to test the job response time of the VM platform in high-workload and low-workload states by using the fault tolerance method based on workload consolidation. As shown by Fig. 8, the job response time in low-workload conditions is approximately 10 s, then increases to approximately 15 s in high-workload conditions. At workload phases 25 and 40, the job response times are increased, which imply a system slowdown. This phenomenon demonstrates that the VM system consumes excessive resources for the fault tolerance mechanism in continuous running state, and issues are further aggravated by the continuous deterioration of overall system performance. The upper limit of server workload expectation is set to 0.8 throughout the fault tolerance. At workload phase 25, fault frequency is reduced and job response time is shortened, thus increasing system service capability.

In this experiment, the prediction accuracy of the VM workload and the reliability of VMs based on the workload consolidation model have been verified. Three methods for resource request allocation are compared. The experimental results have clearly demonstrated that Workload_Consolidation not only improves the reliability and service capability of the VM platform system, it also reduces the fault tolerance frequency of the VM platform with relatively low system cost. The proposed method can very well protect the service capability of the VM platform system.

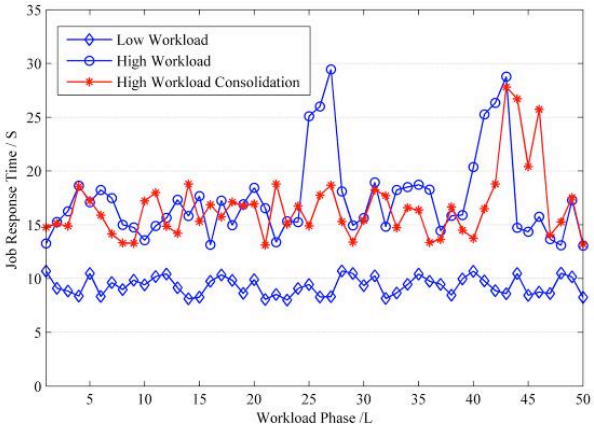


**Fig. 8.** Job response time

## 5. Conclusions

The VMs in cloud platforms experience overload over time and encounter fault tolerance. To protect the reliability of VMs, a fault tolerance method based on the workload consolidation model of VMs in Xen cloud platform was proposed. The method was used to increase VM reliability in cloud platforms and improve the resource efficiency and service capability of VMs. The following conclusions could be drawn:

(1) Workload consolidation in cloud platforms can well optimize the reliability and task processing capability of VMs. The higher the VM workload is, the lower the reliability of VMs and the longer the task processing time will be. Therefore, VMs workload can change the reliability and task processing capability of VMs in cloud platforms.

(2) A VM workload consolidation model is established based on the relationships of VM workload and VM reliability and task processing capability. The model can solve the reliability problems of cloud platforms caused by VM workload and accurately monitor resources. Workload consolidation and allocation prevent the overuse of VMs in cloud platforms.

(3) The fault tolerance method based on the VM workload consolidation model reduces fault tolerance frequency in cloud platforms, optimizes fault tolerance time, and shortens job response time, thus further increasing the reliability and task completion rate of VM platforms.

The proposed method performs fault tolerance on the basis of the workload consolidation of VMs in cloud platforms. The method can be applied to dynamic environments such as VMs in cloud platforms, and it can improve the overall performance of cloud platforms. Consequently, the method can provide convenient and accurate technological support to the fault tolerance of VMs. However, the network I/O resource sharing of VMs is neglected when VM workloads are measured. Thus, studying this gap may increase the applicability of fault tolerance to large-sized network communication VMs.

**Acknowledgements**

The authors are grateful for the support provided by the Key Program for Science and Technology Development of Jilin Province of China (Grant No. 20130206052GX).



## References

1. Goldberg, R. P., "Survey of virtual machine research". *Computer,* 7(6), 1974, pp.34-45.
2. Zhang, Q., Cheng, L., and Boutaba, R. "Cloud computing: state-of-the-art and research challenges". *Journal of Internet Services and Applications,* 1(1), 2010, pp.7-18.
3. Garg, S. K., Toosi, A. N., Gopalaiyengar, S. K., and Buyya, R., "SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter". *Journal of Network and Computer Applications,* 45, 2014, pp.108-120.
4. Xiao, Z., Song, W., and Chen, Q., "Dynamic resource allocation using virtual machines for cloud computing environment". *IEEE Transactions on Parallel and Distributed Systems,* 24(6), 2013, pp.1107-1117.
5. Gómez, A., Carril, L. M., Valin, R., Mouriño, J. C., and Cotelo, C., "Fault-tolerant virtual cluster experiments on federated sites using BonFIRE". *Future Generation Computer Systems,* 34, 2014, pp.17-25.
6. Kakadia, D., Kopri, N., and Varma, V., "Network-aware virtual machine consolidation for large data centers". *Proceedings of the Third International Workshop on Network-Aware Data Management*, New York, USA: ACM, 2013, pp.1-8.
7. Ardagna, C. A., Asal, R., Damiani, E., and Vu, Q. H., "From security to assurance in the cloud: a survey". *ACM Computing Surveys (CSUR),* 48(1), 2015, pp.1-50.
8. Emeras, J., Varrette, S., and Bouvry, P., "Amazon elastic compute cloud (ec2) vs. in-house hpc platform: a cost analysis". *9th International Conference on Cloud Computing (CLOUD)*, San Francisco, USA: IEEE, 2016, pp.284-293.
9. Prodan, R., and Sperk, M., "Scientific computing with google app engine". *Future Generation Computer Systems,* 29(7), 2013, pp.1851-1859.
10. Gulati, A., Holler, A., Ji, M., Shanmuganathan, G., Waldspurger, C., and Zhu, X., "Vmware distributed resource management: Design, implementation, and lessons learned". *VMware Technical Journal,* 1(1), 2012, pp.45-64.
11. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., and Warfield, A., "Xen and the art of virtualization". *ACM SIGOPS operating systems review*, 37(5), 2003, pp.164-177.
12. Binu, A., and Kumar, G. S., "Virtualization techniques: a methodical review of XEN and KVM". *International Conference on Advances in Computing and Communications*, Heidelberg, DE: Springer, 2011, pp.399-410.
13. Ganesh, A., Sandhya, M., and Shankar, S., "A study on fault tolerance methods in cloud computing". *International Advance Computing Conference Computing*, Gurgaon, India: IEEE 2014, pp.844-849.
14. Xu, J.W., Zhang, W.B., Wang, T., Tao, H., "A Genetic Algorithm Based Adaptive Strategy for Image Backup of Virtual Machines". *Chinese Journal of Computers*, 39(2), 2016, pp.351-363.
15. Cully, B., Lefebvre, G., Meyer, D., Feeley, M., Hutchinson, N., and Warfield, A., "Remus: High availability via asynchronous virtual machine replication". *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, San Francisco, USA: ACM, 2008, pp.161−174.
16. Machida, F., Kawato, M., and Maeno, Y., "Redundant virtual machine placement for fault-tolerant consolidated server clusters". *Network Operations and Management Symposium (NOMS),* Osaka, Japan: IEEE, 2010, pp.32-39.
17. Salfner, F., Lenk, M., and Malek, M., "A survey of online failure prediction methods". *ACM Computing Surveys (CSUR),* 42(3), 10, 2010. pp.1-42.
18. Yao, L., Wu, G., Ren, J., Zhu, Y., and Li, Y., "Guaranteeing fault-tolerant requirement load balancing scheme based on VM migration". *The Computer Journal,* 57(2), 2013, pp.225-232.
19. Zhang, Z., Xiao, L., Zhu, M., and Ruan, L., "Mvmotion: a metadata based virtual machine migration in cloud". *Cluster Computing,* 17(2), 2014, pp.441-452.
20. Dong, J., Jin, X., Wang, H., Li, Y., Zhang, P., and Cheng, S., "Energy-saving virtual machine placement in cloud data centers". *13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, Delft, Netherlands: IEEE, 2013, pp.618-624.
21. Asberg, M., Forsberg, N., Nolte, T., and Kato, S., "Towards real-time scheduling of virtual machines without kernel modifications". *Emerging Technologies & Factory Automation (ETFA),* Toulouse, France: IEEE, 2011, pp.1-4.
22. Zhang, J.H., Zhang, W.B., Xu, J.W., Wei, J., Zhong, H., Huang, T., "Approach of virtual machine failure recovery based on hidden Markov model". *Journal of Software*, 25(11), 2014, pp.2702-2714.
23. Mallick, S., Hains, G., and Deme, C. S., "A resource prediction model for virtualization servers". *High Performance Computing and Simulation (HPCS),* Madrid, Spain: IEEE, 2012, pp.667-671.
24. Bruneo, D., Distefano, S., Longo, F., Puliafito, A., and Scarpa, M., "Workload-based software rejuvenation in cloud systems". *IEEE Transactions on Computers,* 62(6), 2013, pp.1072-1085.
25. Wu, Y., Gang, H., Ying, Z., Xiong, Y., and University, P., "A model-based fault tolerance mechanism development approach for cloud computing". *Journal of Computer Research and Development*, 53(1), 2016, pp.138-154.
26. Heath, T., Martin, R. P., and Nguyen, T. D. Nguyen., "Improving cluster availability using workstation validation". *ACM SIGMETRICS Performance Evaluation Review*, 30(1), 2002, pp.217-227.
27. Sahoo, R. K., Squillante, M. S., Sivasubramaniam, A., and Zhang, Y., "Failure data analysis of a large-scale heterogeneous server environment". *Dependable Systems and Networks,* Florence, Italy: IEEE, 2004, pp.772-781.
28. Schroeder, B., and Gibson, G., "A large-scale study of failures in high-performance computing systems". *IEEE Transactions on Dependable and Secure Computing,* 7(4), 2010, pp.337-350.