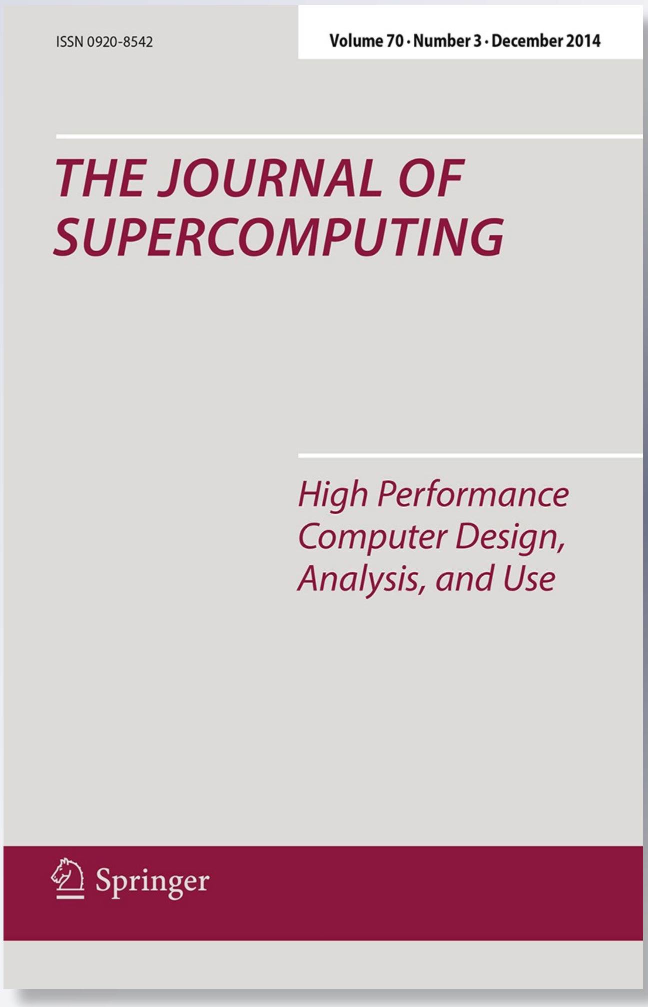
Dynamic forecast scheduling algorithm for virtual machine placement in cloud computing environment

Zhuo Tang, Yanqing Mo, Kenli Li & Keqin Li

The Journal of Supercomputing
An International Journal of High-Performance Computer Design, Analysis, and Use

ISSN 0920-8542
Volume 70
Number 3

J Supercomput (2014) 70:1279-1296
DOI 10.1007/s11227-014-1227-5


ISSN 0920-8542
Volume 70 · Number 3 · December 2014
THE JOURNAL OF SUPERCOMPUTING
High Performance Computer Design, Analysis, and Use
Springer

J Supercomput (2014) 70:1279–1296
DOI 10.1007/s11227-014-1227-5

Dynamic forecast scheduling algorithm for virtual machine placement in cloud computing environment

Zhuo Tang · Yanqing Mo · Kenli Li · Keqin Li

Published online: 17 June 2014

Abstract In most cloud computing platforms, the virtual machine quotas are seldom changed once initialized, although the current allocated resources are not efficiently utilized. The average utilization of cloud servers in most datacenters can be improved through virtual machine placement optimization. How to dynamically forecast the resource usage becomes a key problem. This paper proposes a scheduling algorithm called virtual machine dynamic forecast scheduling (VM-DFS) to deploy virtual machines in a cloud computing environment. In this algorithm, through analysis of historical memory consumption, the most suitable physical machine can be selected to place a virtual machine according to future consumption forecast. This paper formalizes the virtual machine placement problem as a bin-packing problem, which can be solved by the first-fit decreasing scheme. Through this method, for specific virtual machine requirements of applications, we can minimize the number of physical machines. The VM-DFS algorithm is verified through the CloudSim simulator. Our experiments are carried out on different numbers of virtual machine requests. Through analysis of the experimental results, we find that VM-DFS can save 17.08 % physical machines on the average, which outperforms most of the state-of-the-art systems.

Keywords Bin packing · Dynamic scheduling · Forecast · Virtualization · Virtual machine placement

Z. Tang (✉) · Y. Mo · Kenli Li · Keqin Li
College of Information Science and Engineering, Hunan University, Changsha 410082, China
e-mail: ztang@hnu.edu.cn

Keqin Li
Department of Computer Science, State University of New York, New Paltz, NY 12561, USA
e-mail: lik@newpaltz.edu

1 Introduction

Cloud computing is a popular trend in current computing, where people can easily access computational resources in a cheap way. Compared with the previous paradigms, cloud computing focuses on treating computational resources as measurable and billable utilities based on the pay-as-you-go model [1]. From a client point of view, cloud computing can provide an abstraction of the underlying hardware architecture, which can reduce the number of middleware and save the cost of datacenter maintenance. Whereas for cloud providers, they prefer to gain more benefits by hosting more virtual machines (VMs). In this way, the technique of infrastructure as a service (IaaS) focuses on providing a computing infrastructure that leverages system virtualization to allow multiple virtual machines to be consolidated into one physical machine (PM) [2,5]. OpenStack and CloudStack, or other cloud platforms, can support multiple virtualization products such as KVM and Xen [3,4]. For these platforms, the live migration [6,7] technique can ensure that a virtual machine (VM) process does not stop when it is migrated from one computing node to another.

At present, there are various studies focusing on resource scheduling and management, which are often based on load balancing [8,9], dynamic placement [10–12], energy conservation [13,14], and resource classification [15,16]. Considering dynamic change of CPU consumption for virtual machines, Bobroff et al. [17] proposed a dynamic forecast model, which can forecast over an interval shorter than the time scale of demand variability. Those scheduling policies can work well under a stable system environment, for example, constant user requests and fixed infrastructure resources. However, few researchers consider the dynamic change of memory consumption for virtual machines. In most current datacenters, after initialization of memory quotas for virtual machines, the mapping will not be recomputed for the whole process. To make full use of cloud server memories, this paper proposes a dynamic forecast scheduling algorithm called virtual machine dynamic forecast scheduling (VM-DFS), whose framework is shown in Fig. 1.

The VM-DFS algorithm is designed to deploy virtual machines according to the over pre-allocated physical machine memory, which can cause the total consumption of the virtual machines exceeding the physical memory capacity, because the memory spaces in virtual machines are always changing and overlapped. Before deploying virtual machines, the historical memory consumption of each virtual machine will be analyzed, to select the most suitable physical machine to place the running virtual machine according to the future consumption forecast. The bin-packing model is an effective way to optimize the placement of virtual resources [18]. In our algorithm, the virtual machine placement problem is also model a bin-packing problem, and can be solved by the first-fit decreasing (FFD) scheme [19]. In this model, after estimation and adjustment through the memory over commitment techniques for physical machines, such as the VMware ESX Server [22] and KVM virtualization [23], a physical machine can be powered off or set at lower power when there is no virtual machine placed. As shown in Fig. 1, the PM_3 happens to be in a low power state.

The rest of this paper is organized as follows. Section 2 analyzes the problem of virtual machine placement. Section 3 proposes a forecast model for the probability distribution of future memory requirement according to the historical memory usage.

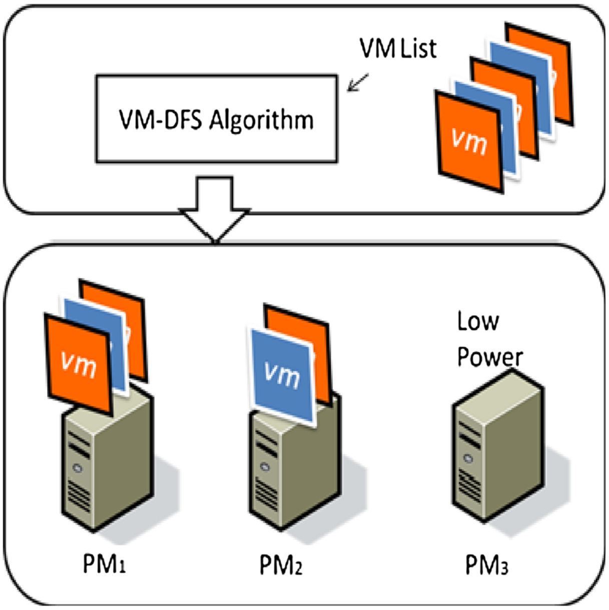


Fig. 1 The VM-DFS placement model

Section 4 formalizes this problem as a bin-packing problem, and gives our dynamic virtual machine placement algorithm VM-DFS. Section 5 presents experiments and analysis which support our contributions. Section 6 concludes this paper.

2 The virtual machine placement problem

The improvement of server resource utilization is an effective way to achieve high performance of datacenters. Through virtualization, multiple virtual machines can be deployed into one physical machine. However, in the current server consolidation process of most datacenters, the virtual machines' quota cannot be changed after they are initialized, despite that the allocated memory resources are not efficiently utilized, thus causing waste of system resources. On the other hand, once the system resources are distributed to specific virtual machines, even though they are not fully used, they are not allowed to be taken advantaged by other virtual machines. Supposing there are n virtual machines deployed on a physical machine PM_j, whose memory capacity is denoted as M_j^c. Every virtual machine VM_i has initial memory $w(i)$, $i = 1, \dots, n$, and the total initial applied memory on PM_j can be calculated as

$$MemApply_j = \sum_{i=1}^{n} w(i).$$

In this model, the actual memory consumption is denoted as MAC_j to quantize the actual usage of all virtual machines in the current physical machine PM_j. At time t, the

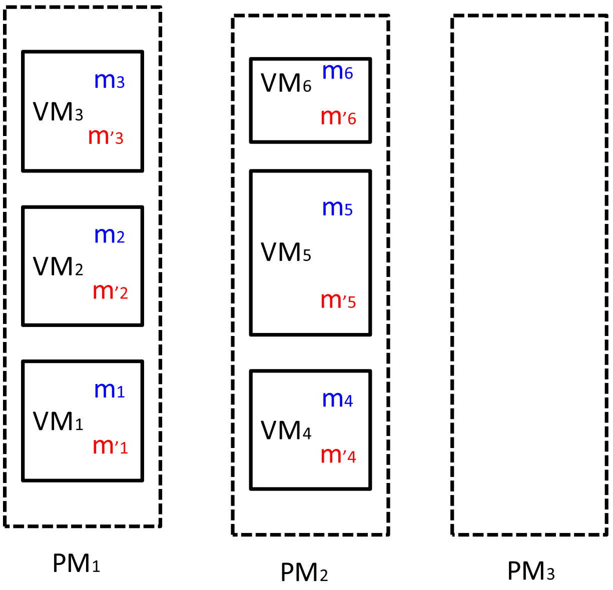


Fig. 2 Virtual machine memory resource allocation and consumption

actual memory consumption of each virtual machine VM_i is denoted as $U_i(t)$. Thus, the summation of memory consumption of all virtual machines can be calculated as

$$MAC_j = \sum_{i=1}^{n} U_i(t).$$

As mentioned above, a virtual machine does not always use all pre-allocated memory. The remainder memory can be recorded as $MemApply_j - MAC_j$. Let us consider the virtual machine placement scenario shown in Fig. 2, where PM_1, PM_2, and PM_3 are the physical machines, each has memory capacity M_j^c, $j = 1, 2, 3$. In this model, assuming that virtualization itself needs to consume some memory resources, and the memory reserved by a physical machine is represented as *MR*. In the initial state, three virtual machines VM_1, VM_2, VM_3 are deployed on the physical node PM_1, and they have initial memory capacities m_1, m_2, m_3. Similarly, PM_2 is deployed with three other virtual machines VM_4, VM_5, VM_6 with initial memory capacities m_4, m_5, m_6. PM_3 is in a low power state. At certain time, the actual memory resource consumption of virtual machine VM_i can be denoted as m_i', $i = 1, 2, 3, 4, 5, 6$.

To have a more thorough understanding of this issue, we specially consider a single physical machine with one virtual machine VM_i placed for the sake of simplicity. Actually, the memory consumption of this virtual machine is dynamically changed. Figure 3 shows $U_i(t)$, i.e., the memory consumption of a live virtual machine VM_i (in terms of the percentage of the memory capacity M_j^c of a physical machine PM_j) as a function of time t. For the static allocation policy, the virtual machine's memory quota is equal to L_m (in percentage of M_j^c), as shown in Fig. 3. In dynamic allocation,

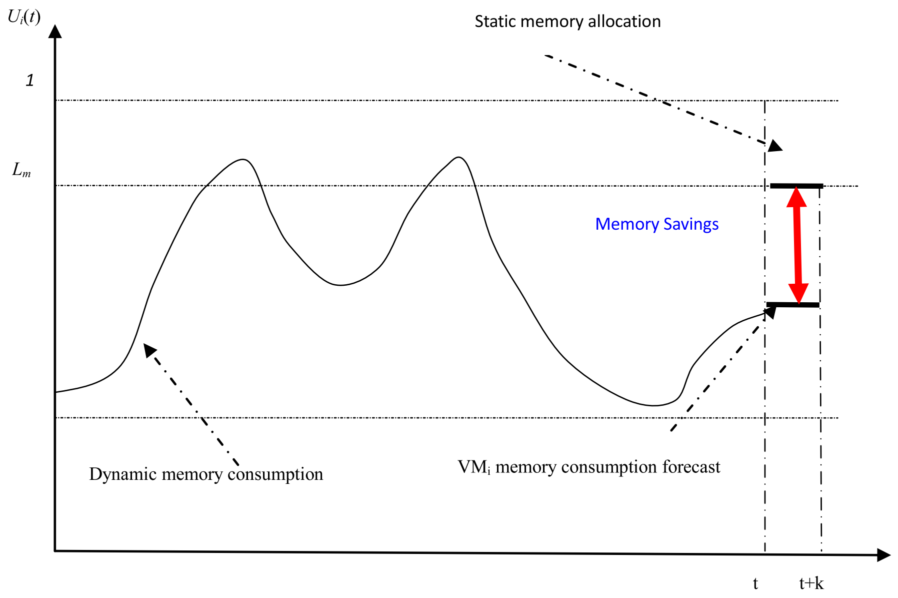


Fig. 3 The change of memory usage by a virtual machine VM_i

the memory size to be allocated for the next interval of length k is predicted at time t, which is based on the virtual machine's historical memory consumption record. Figure 3 shows this whole process clearly. Through an allocation policy with dynamic forecast, the memory saving is

$$MemSaving = L_m - U_i(t + k). \tag{1}$$

In this paper, the memory consumption $U_i(t)$ of a virtual machine VM_i is treated as a random variable with a probability density function (pdf) $u_t(x)$, where $0 \le x \le 1$. The static memory allocation is denoted as L_m. The predicted pdf is $u_{t+k}(x)$, as shown in Fig. 4. Therefore, the average memory saving (in percentage) can be calculated as

$$E[MemSaving] = \int_0^1 (L_m - x) \times u_{t+k}(x)dx. \tag{2}$$

The above equation can be simplified by the following calculation:

$$E[MemSaving] = L_m - \int_0^1 x \times u_{t+k}(x)dx \approx L_m - E[U_i(t)], \tag{3}$$

where we notice that

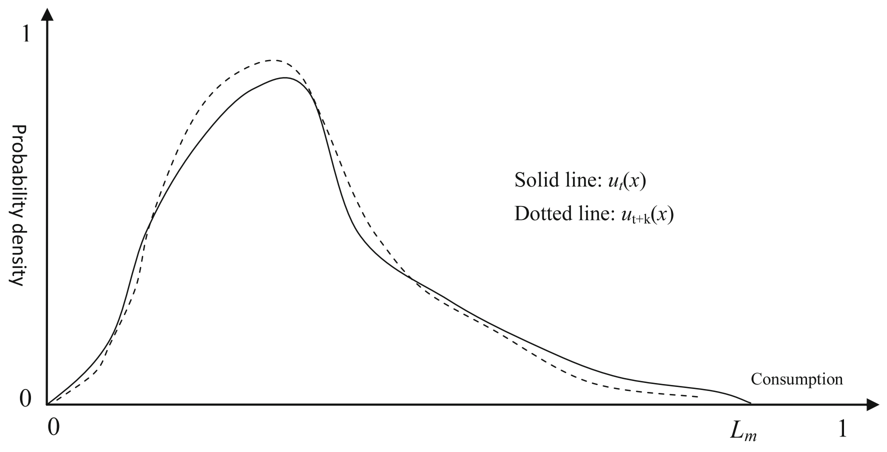


Fig. 4 Probability distribution of virtual machine memory consumption

$$\int_0^1 x \times u_{t+k}(x)dx \approx \int_0^1 x \times u_t(x)dx = E[U_i(t)]. \tag{4}$$

Obviously, the expectation of the prediction value is equal to the mean of the demand distribution, because the prediction is an unbiased estimation. This analysis result can provide a basis for our virtual machine dynamic forecast scheduling algorithm.

Typically, the actual load of a system relates to the running applications, and for virtual machines, the actual memory consumptions are constantly changing. Through the above analysis, for each virtual machine, the pre-allocated memory resource is not fully used at all time. Obviously, a static memory allocation policy will lead to waste of memory resources. If we can take advantage of the idle memory resources effectively and adequately, the overload of the servers will be alleviated accordingly, and the performance and efficiency of the cloud datacenters can also be improved drastically.

This paper proposes a novel virtual machine deployment model. Through employing the predict theory [20], this model reduces overly pre-allocated memory resource of physical machines to virtual machines according to the requirements of the users.

To ensure that all virtual machines on a physical node can meet the memory service level agreement (SLA), the memory of each physical machine will set a threshold value L_m. The threshold can avoid virtual machine memory consumption occasionally coming to a peak, which will cause system instability. This paper introduces a scaling factor to ensure the overload memory consumption as small as possible in the whole-system life cycle. Section 4 gives the definition of this factor as the percentage of the extra virtual machine memory forecast that exceeds the L_m of the physical node.

According to the time-dependent virtual machine memory consumptions, a method is proposed to predict the memory utilization of each virtual machine. The prediction can reflect the variation of the resource usage, which can provide some valid suggestions for virtual machine deployment. The purpose is to minimize the number of servers on the premise, such that the user requirements for memory can all be satisfied, and that a datacenter can reduce the number of physical nodes to the largest extent. The specific predictive model is proposed in the next section.

3 A predictive model

Prediction methods and techniques are widely used for resource allocation and scheduling in distributed systems, including qualitative forecasting, time series analysis, simulation, etc. Shen and Hellerstein [21] decomposed time series into period components, such as daily or weekly variation, and forecasted the request arrival rate of web servers. Bobroff et al. [17] focused on the periodic dynamic change of physical machine CPU in a cloud computing environment. A similar approach is to use the prediction mechanism to make a decision. In light of the theory proposed in [20], through analyzing the historical memory usage, this paper proposes a novel method, which can forecast the probability distribution of future memory demands.

In a time series $t_k, k = 1, 2, \ldots$, the observed value of VM_i's memory consumption is denoted as $U_i(t_k)$. Given a time sequence $t_1, t_2, \ldots, t_j, \ldots, t_k$, we can record the VM_i's observed values as $U_i(t_1), U_i(t_2), \ldots, U_i(t_j), \ldots, U_i(t_k)$.

The virtual machine memory consumption $U_i(t_k)$ is apparently an average value which is acquired from a specific interval, so the time series prediction in this problem can be treated as an auto regression model.

The p-AR [24] model with N samples and n independent variables can be denoted as

$$x_t = \varphi_1 x_{t-1} + \cdots + \varphi_p x_{t-p} + a_t, \tag{5}$$

which can be formalized alternatively as

$$Y = X\beta + \alpha, \tag{6}$$

with

$$X = \begin{pmatrix} x_p & x_{p-1} & \ldots & x_1 \\ x_{p+1} & x_p & \ldots & x_2 \\ \vdots & \vdots & \ddots & \vdots \\ x_{N-1} & x_{N-1} & \cdots & x_{N-p} \end{pmatrix}, \tag{7}$$

and

$$\beta = [\varphi_1, \varphi_2, \ldots, \varphi_p]^T, \tag{8}$$

and

$$\alpha = [a_{p+1}, a_{p+2}, \ldots, a_N]^T, \tag{9}$$

and

$$Y = [x_{p+1}, x_{p+2}, \ldots, x_N]^T, \tag{10}$$

where, Y and α are $(N-p)$-vectors, β is a p-vector, X is an $(N-p) \times p$ matrix, and α refers specifically to the error residuals. To resolve this model, we use the least squares method to estimate the AR coefficient [25]. And for the prediction model AR (2) in this paper, the auto regression can be expressed as

$$U(t+1) = \varphi_1 U(t) + \varphi_2 U(t-1) + \alpha_{t+1}. \tag{11}$$

The parameters φ_1 and φ_2 can be estimated from the statistic data. In this paper, the destination of the least square estimator is to minimize the error residuals, which can be denoted as

$$Min(\varphi) = \sum_{i=1}^{N-p} (x_{p+i} - \varphi_1 x_{p-1+i} - \varphi_2 x_{p-2+i} - \cdots - \varphi_p x_i)^2 = \sum_{i=p+1}^{N} a_i{}^2. \tag{12}$$

As shown in Eqs. (2)–(4), the forecast estimates in the AR model are unbiased, and the residual obeys a Gaussian distribution $N(\mu, \sigma^2)$ [19]. So in this model, the expectation is $E(\mu) = 0$. Beloglazov et al. [14] indicated that when the forecast step length is relatively short, such as two steps, the forecast error is acceptable. Furthermore, the forecast model parameter will be estimated iteratively in the VM-DFS algorithm. In other words, for every virtual machine deployment, the VM-DFS algorithm will estimate the model parameter repeatedly, and the total forecast error will not increase. From the above analysis, in virtual machine deployment, the forecast errors in our algorithm are in an acceptable scope.

4 A dynamic virtual machine placement algorithm

4.1 The bin-packing model for virtual machine placement

In this section, the virtual machine placement problem is formalized as a bin-packing problem. Assuming there are N virtual machines VM_1, VM_2, ..., VM_N. The initial pre-allocated memory quota is W_1, W_2 , ..., W_n. Meanwhile, there are M physical machines PM_1, PM_2, ..., PM_M, with memory sizes M_1^c, M_2^c, ..., M_M^c. The goal is to place the virtual machines into the appropriate physical machines, with the requirement that all virtual machines can obtain the required memory as far as possible, and that the number of used physical machines is minimized. It is well-known that the bin-packing problem is a typical combinatorial optimization problem, and it is also an NP-hard problem [18]. Through formalizing virtual machine placement as a bin-packing problem, we can employ an approximate solution to this problem. For clarity, the correlated variables are listed in Tables 1 and 2.

In our discussion, we define VM_i's memory quota as W_i, and the memory size of the corresponding physical machine as M_j^c. The objective function of this model is to minimize the number of used physical machines. The static deployment model can be described as follows, where the change of memory usage is not considered:

$$\text{minimize } z(y) = \sum_{j=1}^{M} y_i, \tag{13}$$

Table 1 Variable declaration

Variable	Description
N	Number of virtual machines
M	Number of physical machines
W	Initial memory allocation: W_i is the initial memory size of VM_i
M^c	Physical memory capacity: M_j^c is the capacity of PM_j
U	Actual memory consumption: $U_i(t_k)$ is the actual memory consumption of VM_i at time t_k
X	A matrix denoting a mapping of VMs to PMs: $X_{ij} = 1$ iff VM_i is deployed on PM_j
L_m	A threshold in percentage: the maximum amount of PM memory allocated to VMs
r	The percentage of memory forecast exceeding L_m of the PM: $r = (MF - \text{MSLA})/\text{MSLA}$
$VM_i.rm$	The required memory size of virtual machine VM_i
$f_i(t, t+k)$	Predicted memory consumption at time $t+k$ based on prior time t with prediction step k

Table 2 List of abbreviations

Abbreviation	Description
MR	Reserved memory for system itself
MAC	Actual memory consumption
MA	Available memory in one physical machine
MSLA	The allocatable memory satisfying SLA: $\text{MSLA} = L_m \times MA$
MF	The forecast value of total memory consumption in one physical machine

such that

$$\begin{cases} \sum_{j=1}^{M} X_{ij} = 1, \forall i \in \{1, 2, \ldots, N\}; \\ \sum_{i=1}^{N} W_i X_{ij} \leq M_j^c y_j, \forall j \in \{1, 2, \ldots, M\}; \\ X_{i,j} \in \{0, 1\}, \forall i \in \{1, 2, \ldots, N\}, \forall j \in \{1, 2, \ldots, M\}; \\ y_j \in \{0, 1\}, \forall j \in \{1, 2, \ldots, M\}; \end{cases} \tag{14}$$

where X_{ij} in matrix X is a boolean variable which denotes whether the virtual machine VM_i is deployed on the physical machine PM_j. Similarly, y_j indicates whether PM_j is deployed with some virtual machines. The constrains are designed to make sure that each virtual machine can only be placed into one physical machine, and each physical machine has a capacity constraint M_j^c to limit the number of virtual machines deployed on it. The objective is to satisfy the memory requirement of each virtual machine and to minimize the number of physical machines in a cloud datacenter. Obviously, the less physical nodes are used, the more energy resources are saved.

To acquire dynamic prediction of memory usage for virtual machine scheduling and migration, the conditions in the above bin-packing problem should be improved as follows. Since each physical machine has a constraint on maximum memory usage, the memory consumption of each physical node should not exceed the MSLA. For each physical machine, the memory consumptions of virtual machines are always changing dynamically. This paper proposes a novel virtual machine deployment algorithm, which takes the predicted memory consumption MF_j into consideration. The predictor MF_j can be calculated as follows:

$$MF_j = \sum_{i=1}^{N} X_{ij} f_i(t, t+k), \tag{15}$$

where $f_i(t, t+k)$ represents the memory forecasting value at time $t+k$ for a special virtual machine VM_i, and the parameter k denotes the step of the forecast process.

This method can predict the virtual machine memory consumption in the next service cycle, and attempt to make the servers supply the maximal actual memory usage. Meanwhile, the overload percentage MSLA can be decreased to a minimum. We consider the following two cases.

– When the actual memory consumption in a physical server exceeds the preset percentage, the virtual machine migration service should be launched. The appropriate virtual machine should be shutdown or be migrated to another physical node, and the physical server should not be loaded any more virtual machines.
– When the predicted memory consumption for a specific physical server exceeds the preset percentage, no more virtual machine should be deployed on it.

The deployment model using the dynamic prediction method in this paper can provide some suggestions to choose a suitable physical machine for the given virtual machines. This dynamic deployment model based on the bin-packing problem can be described as follows:

$$\text{minimize } z(y) = \sum_{j=1}^{M} y_i, \tag{16}$$

such that

$$\begin{cases} \sum_{j=1}^{M} X_{ij} = 1, \forall i \in \{1, 2, \ldots, N\}; \\ \sum_{i=1}^{n} U_i(t) X_{i,j} \leq L_m M_j^c y_j, \forall j \in \{1, 2, \ldots, M\}; \\ MAC_j = \sum_{i=1}^{N} U_i(t) X_{i,j}, \forall j \in \{1, 2, \ldots, M\}; \\ MF_j = \sum_{i=1}^{N} X_{i,j} f_i(t, t+k), \forall j \in \{1, 2, \ldots, M\}; \\ VM_i.rm < MSLA_j(1+r) - MAC_j, \forall i \in \{1, 2, \ldots, N\}, \forall j \in \{1, 2, \ldots, M\}; \\ (MF_j - MSLA_j)/MSLA_j < r, \forall j \in \{1, 2, \ldots, M\}; \\ X_{i,j} \in \{0, 1\}, \forall i \in \{1, 2, \ldots, N\}, \forall j \in \{1, 2, \ldots, M\}; \\ y_j \in \{0, 1\}, \forall j \in \{1, 2, \ldots, M\}; \end{cases} \tag{17}$$

where the L_m value can be preset according to the system context environment, the variable $U_i(t)$ represents the actual memory consumption of the current virtual machine VM_i, and the variable MF_j can be regarded as the predicted memory consumption of

PM_j, which is based on time series. In addition to the constraints in the static deployment model, the extra constraints in the dynamic deployment model are designed to make sure that the requested memory of a virtual machine does not cause excessive memory consumption on a physical machine, and that the predicted memory consumption does not cause the extra memory consumption to exceed r.

4.2 The VM-DFS algorithm

In this paper, we merely consider memory resource consumption for virtual machine placement. In the VM-DFS algorithm, the first-fit descending (FFD) algorithm [19] is employed to select the most adaptive physical machines for current virtual machines. FFD provides a fast but often non-optimal solution, involving placing each item into the first bin in which it will fit. The algorithm can be made much more effective by first sorting the list of elements into a decreasing order, although this still does not guarantee an optimal solution and for longer lists may increase the running time of the algorithm.

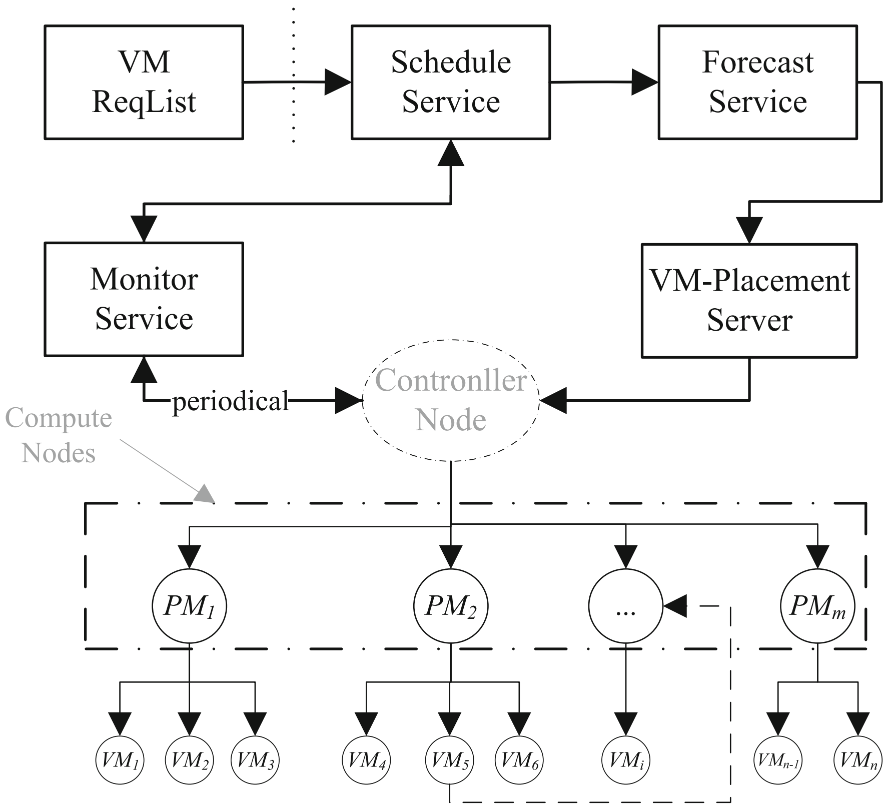


Fig. 5 The architecture for dynamic prediction and VM scheduling

Figure 5 shows the workflow architecture of the VM-DFS algorithm, which is outlined as follows.

- *System initialization* Set the system initial parameters.
- *Physical machine selection* for each virtual machine request, analyze the monitoring management service information first, and then call the prediction service, which can count and analyze the historical virtual machine memory consumption, and finally predict the future memory usage.
- *Virtual machine deployment* Collect all the virtual machine actual memory consumption using the system monitoring service periodically through the first-fit descending algorithm.

The detailed description of VM-DFS is given in Algorithm 1. According to the FFD algorithm, the physical machines are checked to find the first suitable one to place the current virtual machine. We notice that a virtual machine VM_i is deployed on a physical machine PM_j if the requested memory $VM_i.rm$ of the virtual machine does not cause excessive memory consumption on PM_j (lines 7–8). Furthermore, the predicted memory consumption of VM_i does not cause the extra memory consumption to exceed r (lines 9–15).

Algorithm 1 Virtual machine dynamic forecast scheduling (VM-DFS)

```
Require:
    $VMList$: The requested virtual machine list to be placed: $VM_1, VM_2, \ldots, VM_N$;
    $PMList$: The current physical machine list $PM_1, PM_2, \ldots, PM_M$;
    $L_m$: The threshold for the maximum physical machine memory allocated to virtual machines;
    $r$: The percentage of VM memory forecast exceeding $L_m$ of the PM.
Ensure:
    $X$: The matrix denoting the mapping of virtual machines to physical machines.
1: for each $VM_i$ in $VMList$ do
2:   for each $PM_j$ in $PMList$ do
3:     $MA_j = M_j^c - MR_j$; //Calculate the available memory in $PM_j$
4:     $MSLA_j = L_m \times MA_j$; //Calculate the allocatable memory while satisfying SLA
5:     $MAC_j = \sum_{i=1}^{N} U_i(t)X_{ij}$; //Calculate the actual memory consumption in $VM_j$
6:     $MF_j = \sum_{i=1}^{N} X_{ij} f_i(t, t+k)$; //Forecast the total consumption of all virtual machines in $VM_j$
7:     //Determine whether $VM_i$ can be deployed on $PM_j$
8:     if $VM_i.rm < (MSLA_j \times (1+r) - MAC_j)$ then
9:       //Calculate the percentage of forecast memory consumption exceeding $MSLA_j$
10:      $r' = (MF_j - MSLA_j)/MSLA_j$
11:      //Determine whether the future requirement of $VM_i$ can be satisfied on $PM_j$
12:      if $r' \leq r$ then
13:        $X_{ij} = 1$; //Place $VM_i$ on $PM_j$
14:        break; //Continue to place the next $VM_{i+1}$
15:      end if
16:    end if
17:  end for
18: end for
19: return
```

5 Experimental performance analysis

The experimental environment is based on CloudSim [26], which is an open source cloud simulation toolkit. We extend the CloudSim platform to implement the algorithm in this paper. Some components and classes in source codes such as "datacenter" and "host" have been modified to fit the program running. In this experiment, the virtual machine list with memory quota is initially defined as (4, 2, 3, 2, 1, 3, 3, 3, 1, 4, 1, 4, 2, 3, 3, 4, 3, 1, 2, 4), whose dimension is recorded as Gigabyte. In this list, the total memory size is 53 GB, and for each physical machine, the inherent memory resource MA is preset as 12 GB, and the reserved memory resource MR can be set to 2 GB. In [14], the threshold depends on the allocated resource it is using, and ranges from 0.75 to 0.90. Considering that the threshold should be set lower if the resource consumption changes frequently, to satisfy the virtual machine requirement under the memory SLA, the threshold L_m is set to 0.8 in this experiment environment. Meanwhile, for the actual virtual machine memory consumption, we denote the percentage of extra memory as variable r, and its value is related to the following specific experimental context.

This paper conducts the experiments in the following aspects.

Scenario 1 Comparison of virtual machine deployment between the algorithm in this paper and the default VM-static algorithm in CloudSim.

Scenario 2 Comparison of numbers of physical nodes under different numbers of virtual machines with different memory quota orderings.

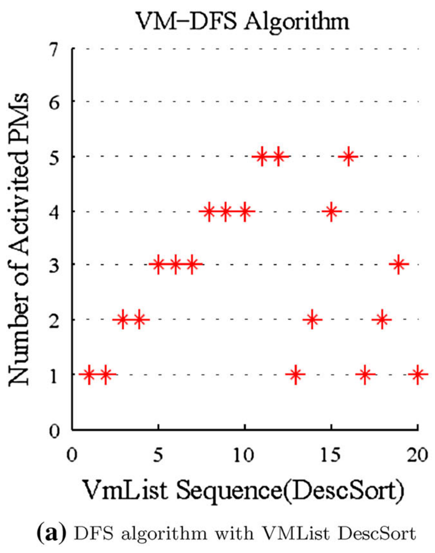
Scenario 3 Comparison of the impact of memory over-consumption on the number of activated physical machines.

In these experiments, the VM-DFS algorithm is compared with other virtual machine placement policies. For simplicity, in the static algorithms, the actual memory consumption of each virtual machine is not taken into account. The memory consumption changes of the virtual machines are generated through the random function in CloudSim. To improve the accuracy of experimental results, each group of experiment is run multiple times, and we finally choose the average results.

5.1 Experiment of virtual machine distribution

In this experiment, we set the value r (i.e., the percentage of extra memory consumption) to 0.1. From simulation and experimentation results, we can see that at least 6 physical machines are required when using the static deployment method. However, by the advantage of dynamic forecast, only 5 physical machines are needed when the VM-DFS algorithm is employed. The main reason is that the scheduling program will first forecast the memory consumption before the deployment of each virtual machine, according to the definition $r' = (MF - MSLA)/MSLA$. Meanwhile, the memory quota of the virtual machine list in this process can be sorted in a non-increasing order (4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 1, 1, 1, 1).

As shown in Fig. 6, at the beginning of deploying stage, since there is no virtual machine being deployed on any physical machine, the prediction mechanism has not yet begun to work. The results indicate that, for the last 3 virtual machines, the VM-DFS algorithm does not need to start any more physical machine comparing with the



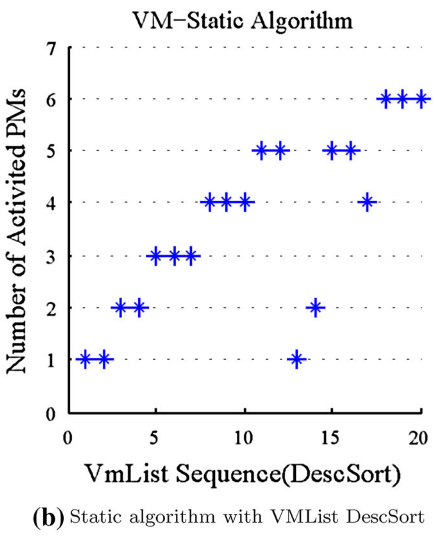


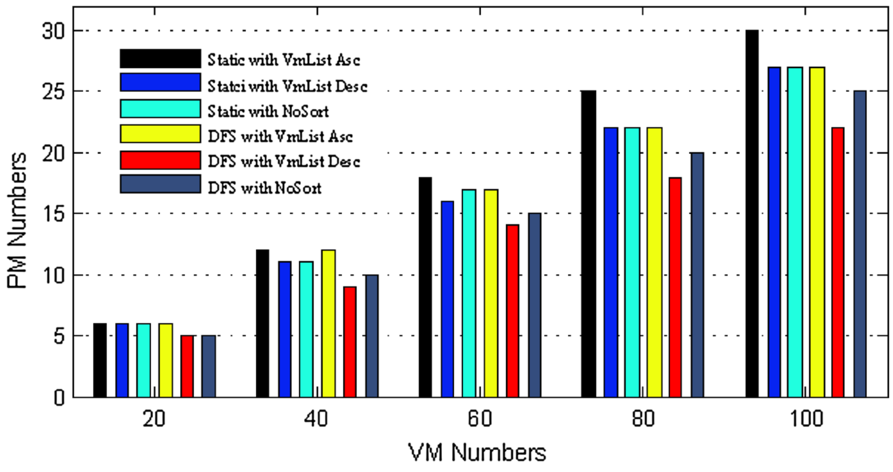
Fig. 6 Comparison of virtual machine distribution between DFS and VM-static algorithms

static algorithm. Because at the beginning, the deployment policy prefers to choose the large memory quota. In this experiment, we can learn that the average number of the physical machines in demand is 5, and the number is 6 for the static algorithm. From this experiment, by saving more physical hosts and using the memory resources more effectively, the VM-DFS algorithm can get better performance than other static algorithms.

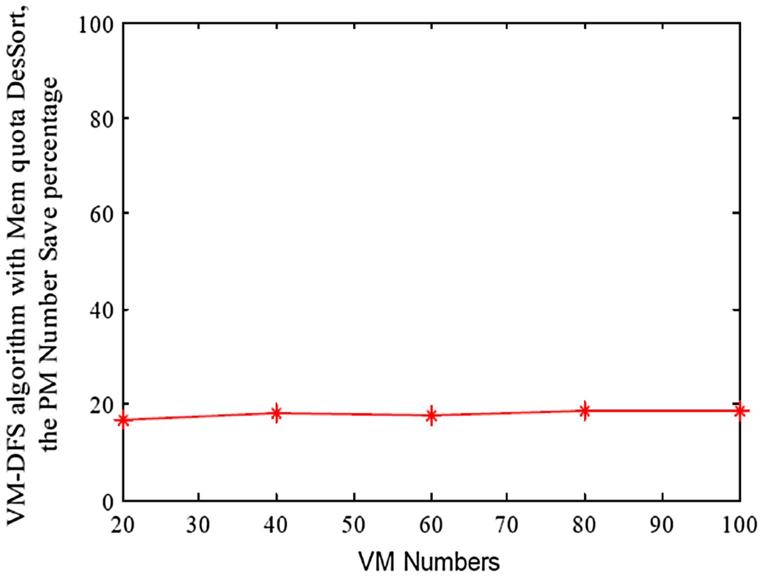
5.2 Experiment for different orders of memory quotas

To verify the stability and scalability of the VM-DFS algorithm, we change the virtual machine list in the experiment, with different orderings of the virtual machine memory quota, i.e., ascending order, descending order, and random order. The memory quota of each virtual machine is generated from a random function, which is in the scope of $\{1, 2, 3, 4\}$.

To improve the accuracy of experimental results, each set of experiments was run multiple times, finally the average number of all results is counted as the final outcome.From the experiment results shown in Fig. 7a, we can learn that the VM-DFS algorithm with a descending order of quota in the virtual machine list can get the best performance. There is no doubt that the VM-DFS algorithm acquires better performance than the default static scheduling algorithm in CloudSim. With the virtual machine number increasing, the VM-DFS algorithm can save more physical machines. Through calculating the results in Fig. 7a, the average percentage of saved physical machines in each experiment is shown in Fig. 7b. We also calculate the average values of the 5 experiments in Fig. 7b with different numbers of virtual machines (20, 40, 60, 80, and 100), and get the average percentage of saved physical machines, which can reach 17.08 %.



(a) The number of physical machines used



(b) The percentage of saved physical machines

Fig. 7 The saving of PMs using the DFS algorithm

5.3 Experiment on the impact of over-consumed memory

In this experiment, the number of physical machines used by our VM-DFS algorithm is compared with that of the static virtual machine deployment (VM-static) algorithms for different memory over-consumption percentages r. Because the pre-allocation of memory is usually greater than the allocatable memory in a physical machine, in these experiments, the value of r is larger than zero when the total memory consumption of

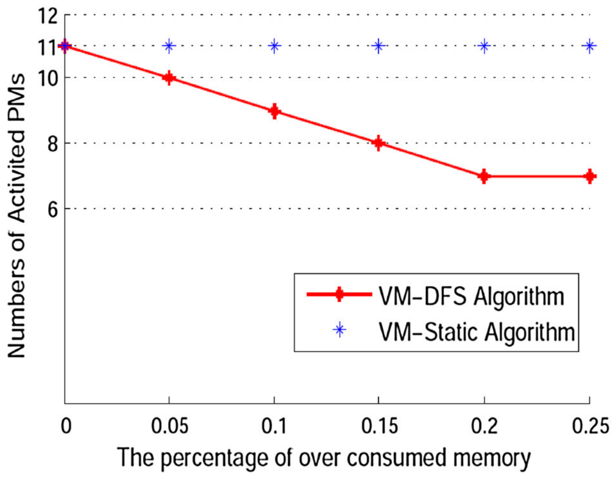


Fig. 8 Number of PMs vs. memory over-consumption percentage

the virtual machines exceeds the MSLA of the physical machine. As shown in Fig. 8, where the number of the virtual machines is changed to 40, the number of physical machines reaches 11, if the memory over-consumption percentage in the VM-DFS algorithm is close to zero. When $r \geq 0.15$, the VM-DFS algorithm can save at least 3 physical nodes. When $r \geq 0.2$, the VM-DFS algorithm can save at least 4 physical nodes. But in an actual environment, some other constraints should be considered. For example, when the memory consumption of some virtual machines come to peak, the system should migrate some virtual machines to other physical nodes at once or close some idle virtual machines.

In general, virtual machine migration will degrade the performance of the physical machines. Therefore, we can learn that if we pre-allocate more memory for the virtual machines, the physical system performance will be more degraded. Through the VM-DFS algorithm, some policies of virtual machine placement can be supplied for a cloud datacenter, which can implement the energy efficiency to some extent.

6 Conclusions

This paper proposes a novel virtual machine deployment algorithm for a cloud computing environment, which is based on the excessive pre-allocated physical machine memory. In this model, virtual machine placement is formalized as a bin-packing problem, which can be solved by employing the FFD scheme. Especially, based on dynamic deployment and time series forecasting theory, the p-AR's coefficient parameters can be estimated in our model through least square. Based on the bin-packing model and a dynamic forecast method, the proposed algorithm can minimize the number of active physical machines relatively. Furthermore, every virtual machine can conform to the required memory SLA. In future research, more system resources such as CPU and storage will be considered in our model, so that our method can be applied to more realistic situations in a cloud computing environment.

Acknowledgments The authors are grateful to the three anonymous reviewers for their criticism and comments which have helped to improve the presentation and quality of the paper. This work is supported by the Key Program of National Natural Science Foundation of China (Grant No. 61133005), National Natural Science Foundation of China (Grant Nos. 61103047 and 61370095), and Open Foundation of State Key Laboratory of Software Engineering (SKLSE20 I2-09-18).

References

1. Armbrust M, Fox A, Griffith R, Joseph AD et al (2010) A view of cloud computing. Commun ACM 53(4):50–58
2. Barham P, Dragovic B, Fraser K et al (2003) Xen and the art of virtualization. ACM SIGOPS Oper Syst Rev 37(5):164–177
3. Zhang F, Sakr MF (2014) Performance variations in resource scaling for mapreduce applications on private and public clouds. In: Proceedings of The 7th IEEE international conference on cloud computing (cloud), Alaska, June 2014
4. Wen X, Gu G, Li Q, Gao Y, Zhang X (2012) Comparison of open-source cloud management platforms: OpenStack and OpenNebula. In: Proceeding of IEEE 9th International Conference on Fuzzy Systems and Knowledge Discovery, pp 2457–2461
5. Padala P, Zhu X, Wang Z, Singhal S, Shin KG et al (2007) Performance evaluation of virtualization technologies for server consolidation. HP Labs Tec. Report, Palo Alto
6. Clark C, Fraser K, Hand S et al (2005) Live migration of virtual machines. In: Proceedings of the 2nd conference on Symposium on Networked Systems Design and Implementation vol 2, pp 273–286
7. Liu H, Jin H, Liao X, Hu L, Yu C (2009) Live migration of virtual machine based on full system trace and replay. In: Proceedings of the 18th ACM international symposium on High performance distributed computing, pp 101–110
8. Pearce O, Gamblin T, de Supinski BR, Schulz M, Amato NM (2012) Quantifying the effectiveness of load balance algorithms. Proceedings of the 26th ACM international conference on Supercomputing, pp 185–194
9. Singh A, Korupolu M, Mohapatra D (2008) Server-storage virtualization: integration and load balancing in data centers. In: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, pp 53
10. Carrera D, Steinder M, Whalley I, Torres J, Ayguadé E (2008) Managing SLAs of heterogeneous workloads using dynamic application placement. In: Proceedings of the 17th international symposium on High performance distributed computing, pp 217–218
11. Verma A, Ahuja P, Neogi A (2008) Power-aware dynamic placement of HPC applications. In: Proceedings of the 22nd annual international conference on Supercomputing, pp 175–184
12. Lucas Simarro JL, Moreno-Vozmediano R, Montero RS, Llorente IM (2011) Dynamic placement of virtual machines for cost optimization in multi-cloud environments. 2011 International Conference on High Performance Computing and Simulation (HPCS), pp 1–7
13. Chen G, He W, Liu J, Nath S, Rigas L, Xiao L, Zhao F (2008) Energy-aware server provisioning and load dispatching for connection-intensive internet services. NSDI, pp 337–350
14. Beloglazov A, Abawajy J, Buyya R (2012) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener Comput Syst 28(5):755–768
15. Stage A, Setzer T (2009) Network-aware migration control and scheduling of differentiated virtual machine workloads. In: Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, IEEE Computer Society, pp 9C–14
16. Tang Z, Zhou J, Li K, Li R (2012) MTSD: a task scheduling algorithm for MapReduce base on deadline constraints. Parallel and Distributed Processing Symposium Workshops and PhD Forum (IPDPSW), 2012 IEEE 26th International, IEEE Computer Society
17. Bobroff N, Kochut A, Beaty K (2007) Dynamic placement of virtual machines for managing sla violations. In: 10th IFIP/IEEE International Symposium on Integrated Network Management, pp 119C–128
18. Coffman EG Jr, Garey MR, Johnson DS (1996) Approximation algorithms for bin packing: a survey. Approximation algorithms for NP-hard problems, pp 46–93
19. Coffman EG Jr, Garey MR, Johnson DS (1991) A simple proof of the inequality $FFD(L) \leq 11/9OPT(L) + 1$, L for the FFD bin-packing algorithm. Acta Math Appl Sinica 7(4):321–331

20. Box GE, Jenkins GM, Reinsel GC (2013) Time series analysis: forecasting and control. Wiley.com, New York
21. Shen D, Hellerstein JL (2000) Predictive models for proactive network management: application to a production web server. In: Proceeding of Network Operations and Management Symposium, pp 833–846
22. Petrovic D, Schiper A (2012) Implementing virtual machine replication: a case study using Xen and KVM. In: Proceedings of 26th international conference on advanced information networking and applications(AINA), pp 73–80
23. Li C-H (2012) Evaluating the effectiveness of memory overcommit techniques on kvm-based hosting platform. In: Proceedings of World Academy of Science, Engineering and Technology, no. 70, World Academy of Science, Engineering and Technology
24. Dickey DA, Fuller WA (1979) Distribution of the estimators for autoregressive time series with a unit root. J Am Stat Assoc 74(336a):427–431
25. Schneider T, Neumaier A (2001) Algorithm 808: ARfit Matlab package for the estimation of parameters and eigenmodes of multivariate autoregressive models. ACM Trans Math Softw 27(1):58–65
26. Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41(1):23–50