A self-aware strategy for virtual machines placement on clouds

Fen Guo[1*], Huaqing Min[1], Ming Yin[2]

[1]*School of Software Engineering, South China University of Technology, Guangzhou Higher Education mega center, 510006, Guangzhou, China*

[2]*School of Automation, Guangdong University of technology, Guangzhou Higher Education mega center, 510006, Guangzhou, China*

Received 1 August 2014, www.cmnt.lv

Abstract

Cloud computing is a new computing service mode, and virtualization is a key technology of it. A self-aware strategy (SAST) for Virtual machines (VMs) management on clouds is proposed which is multi-attributed weighted on the resources. It manages the virtual resource basing on the requests of users and the real-time state of the system dynamically. It consists of three phases: (1) monitoring the cloud performance including VMs and Physical Machines (PMs), with the data standardized; (2) measuring the cloud load balance value with the attribute weighted measurement model; (3) using the placement algorithm to choose the best appropriate PM to place the VM requested. The main contribution of the paper is that a cloud load balance measurement model is introduced and a VM scheduling strategy is proposed which includes the VMs placement optimization algorithm and the VMs dynamic migration algorithm. The SAST is tested on the simulation platform comparing with other traditional ones. As a result, we concluded that it guaranteed the SLA and achieved better load balance of cloud. And at the same time, it minimized the number of the started PMs on clouds to reduce energy consumption.

Keywords: cloud computing, virtualization, placement, scheduling

1 Introduction

Cloud computing [1] is considered a new computing service model and has got a lot of attentions. Cloud computing provides the infinite shared resources to the customers via the Internet. The customers would get them with not understanding the fundamental technology and the application running environment. The more quickly the demand of cloud computing grow, the larger data centre is needed. As most of all know, the user requirement for cloud platform is often heterogeneous and irrelevant, so the unreasonable resource distribution will lead to the waste of resources inevitably. In order to improve the utilization rate of resources, cloud computing platform also needs the dynamically balanced load of various kinds of services. On the other hand, large-scale computing infrastructure consumes a lot of power resources, and the power consumption increased year by year. FORREST W [2] has predicted that the data centre power consumption would reach 2% of the world's total energy consumption in 2020. So, how to dynamically and effectively manage the cloud computing platform resource becomes a key problem.

Virtualization technology [3, 4] provides an effective method to manage the resources of the cloud computing platform dynamically, and it has broken the tight coupling between the computing and the hardware. Customers use Virtual Machines (VMs), based on SLA; cloud providers take advantages of VM's flexible management on PMs to optimize resources allocation so as to meet customers' requests. Server virtualization technology enables multiple VMs running on a physical node at the same time. It has greatly improved the utilization of computing resources and implemented the on-demand deployment [5, 6]. In addition, the VM migration technology makes that cloud computing platform can be dynamically adjusted to deploy the VM to the less physical machines according to the changes of the service load, and as a result, it will achieve energy saving.

Since different resource utilization is caused by different mapping between VMs and PMs, for cloud providers, the key issue is how to effectively manage and schedule VMs to meet the customer's requests, and at the same time reduce the energy consumptions to minimize the cost. Nowadays, the algorithm for building the mapping from the customer's requesting VMs sets to the servers in the resource pool is becoming a hot issue. The algorithm will choose the most appropriate PM as the host for the requested VM and establish the specific mapping.

The number of the VMs requested on cloud is increasing with the development of cloud computing, and then the deployment of VMs becomes more important. A VM will be bound a PM for a lifelong under simple managing strategy, which will lead to a load imbalance. In the same way, the managing and scheduling strategy based on single attribute will also cause uneven load, such as a VM with demand for network could be deployed in a PM with sufficient CPU residue but poor network resource, resulting the VMs on the same PM competing for bandwidth. At the same time, the quantity and load of VMs and PMs will vary over time with the demand of the customer and application, static management of VMs will cause waste or shortage of resources, and the artificial scheduling is an obvious lag one.

[*]***Corresponding author*** e-mail: csguofen@scut.edu.cn

For VM management, one major direction is to minimize the number of the PMs for reducing energy consumption. Such problems are usually to be interpreted as packing problem [7-9]. These studies are usually only consider how to reduce the number of physical, but does not take into account the needs of customers (for example, now more and more customers have specified requirements about the network).What's more, it will lead SLA violation. Some studies [10-22] considered the needs of customers and applications. However, the optimization of load balancing is less concerned, and some studies do not consider the dynamic performance of the cloud. In view of this, the VMs management strategy should not only consider the utilization rate of resources but also the overall load balancing combining with the dynamic demand of customers.

Therefore, our paper presents a dynamic managing strategy for VMs on cloud. It is concerned chiefly with meeting the customer requirement, keeping the system load balancing and saving power. With meeting the constraints of PM resources(CPU, network, etc.), the strategy managed the VMs on PMs to achieve cloud's load balancing for reducing SLA violation, which allows the idle PMs in a sleeping state so as to reduce energy consumption on cloud finally.

Based on the above, we proposed a self-aware strategy (SAST) for VMs management on clouds which is called as SAST. The strategy is based on the user requests of virtual resources and the real-time system status. And it is composed of three phases: Firstly, to monitor the cloud load performance including VMs and physical machines, with the data standardized; secondly, to measure the cloud load balancing value with the proposed measurement model. Thirdly, to use the managing and deploying algorithm to choose the best appropriate PM to deploy the VM requested. In addition, the batch requests are treated in accordance with the single application process in order in SAST.

Our paper is divided into six major sections as follows. Section 1 introduces the related work. Section 2 opens with the description of the SAST. Section 3 shows the load balance measurement method. Section 4 introduces the main scheduling algorithm, while section 5 provides experimental result and analysis, and section 6 draws a conclusion.

2 Related work

There are two focuses on VM managing strategy for reducing energy consumptions. One is to consider how to place VM to the PMs in cloud. Eucalyptus [10, 11] has proposed round robin and greed algorithm to deploy the VMs. While OpenStack [12] takes a random scheduling strategy as the default one. However, the match-making scheduler [13] of OpenNebula achieved the ranking algorithm. Varma et al. [14] dynamically readjust server's location and consider the cost of migration and energy, with a simple algorithm; it shows that dynamic migration technology realizes low energy cost. Norman Bobroff et al. in [15] put forward a VM deployment algorithm using the forecasting techniques and heuristic algorithm, and ensured the SLA with minimizing the physical machine number. Singh et al. formed the question into a multi-dimensional knapsack problem, and treated the constraints of the deployment as a separate dimension [16]. Tsakalozos [17] used a two-phase mechanism for mapping problem of VM for large heterogeneous infrastructure: First, to synthesize the physical machine set which can be used at this stage; second, to determine the approximate optimal VM-to-PM mapping with satisfying constraints. Breitgrand et al. formalized the problem to be a multi-unit combinatorial auction model in [18]. Zhang Wei et al. of [19] proposed a kind of strategy for deploying and scheduling VMs based on multiple attributes analysis for the uneven loads between physical machines in the cloud computing.

The strategies or algorithm above consider the energy consumption, and they neglect the user requests and the load balancing of the cloud. The other type [20-22] is to consider the dynamic factor of the cloud. Nik Bessis [20] explored two configurations, the static case in which VMs are generated according to the cloud orchestration, and the dynamic case in which VMs are reactively adapted according to the job submissions, using migration, for optimizing performance time metrics. Yang Xing [21] proposed the Performance Matching-Load Balancing (PM-LB) algorithm of VM deployment. In [22] Zhao Hongwei put forward a kind of efficient resource management strategy based on the domain. Studies above covered various aspects, and they emphasized with the merchant's profits or only one single aspects of the optimization. But the study didn't consider how to achieve best load balancing. So they are also different from our objective.

3 Description of the SAST

3.1. SYSTEM ARCHITECTURE

The framework of the SAST is shown in Figure 1. It includes: central controller; VMs scheduling strategy generator, cloud monitor, VMs placement module.

The central controller is responsible for the overall system, including receiving the requests from the client layer, sending alarm information from the monitor and sending command to the VMs scheduling generator, etc.

The scheduling generator would build the load balance model of the cloud, and judge whether the node is overloaded or low load by analyzing the real-time data from the monitor. And then, it will calculate the best VMs placement scheme using the proposed algorithm in SAST combining with the requests of the alarm queue and the user's optimizing request queue.

The monitor will receive the load performance of each VM and PM periodically, the requests of the migration, the usage of the storage and the network of the cloud. If it found a PM load had exceeded the specified threshold, it would alarm. And at the same time, it would add this PM

into the alarm queue, and informed the central controller triggering the VMs scheduling generator to deal with it.

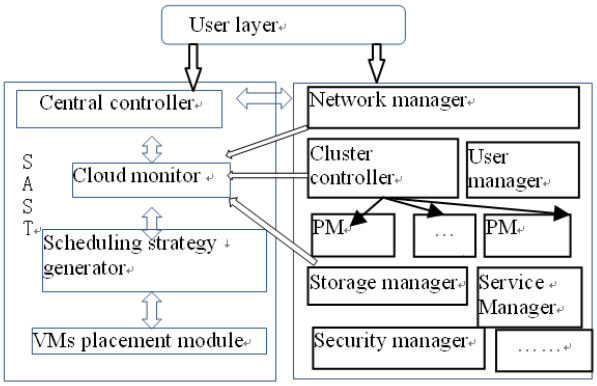

FIGURE 1 The system architecture

3.2 THE OBJECTIVE

The details of our objective are as follows:

Meet the existing customer's request, and maximize the load balancing degree on each physical server scheduling domain.

Minimize the number of PMs to start on IaaS on the basis of 2 to save energy.

Control the load rate of each PM in the scheduling domain not higher than the threshold

3.3 MONITORING STRATEGY

The traditional distributed systems often use the passive monitoring strategy. The node does not only need to exchange information between each other, but also report the status to the information centre regularly. Consequently, it will affect the overall performance of the platform.

1. The monitoring contents are shown in Table 1.

TABLE 1 The monitoring contents

CPU	Memory	Hard disk	Network
1-usage	1-usage	1-usage	1-usage

2. The sampling method.

The average sampling method is used to collect the monitoring data. The monitor collects every node's (including PMs and VMs) load data from the system for every *T* seconds, and defines the average value of the last N data as the load value of the node, which is denoted as $X = \sum_{i=1}^{N} X_i / N$. The value of *T* and *N* can be dynamically set by the monitor according to the individual request, the default value of *T* is 10 seconds, and the *N* is 5.

3. The monitor saves the latest load data vector $\boldsymbol{S}_x$ as the real-time load information of each *X* as follows(*X* can be a VM or a PM).

$$\boldsymbol{S}_x = (C_X, M_X, H_X, N_X). \quad (1)$$

C_x, M_x, H_x, N_x, respectively represents the monitoring data of CPU usage, memory usage, hard disk usage and bandwidth usage of the *X*.

3.4 SCHEDULING STRATEGY

The scheduling strategy is the core of the SAST, and it will be detailed in the below. This scheduling strategy combined with users' requirements to overcome the shortcomings of existing technology:

1) A cloud platform load balance measurement model is introduced with setting the weigh vector and matrix referring to the customer requirements and the monitoring data.

2) A VM scheduling algorithm is proposed, which includes the VM deployment optimization strategy and the VM dynamic migration strategy.

4 Load balance value measurement method

4.1 DEFINITION AND LEMMA

Suppose the number of the PMs in the scheduling domain is *m*, and the one of the VM types is *t.*

Definition 1. Let *P* stands for the PMs Set. *P* is defined as follows.

$$\mathrm{P} = \left\{ \mathrm{G}_i | \mathrm{i} \in (0, n) \, and \, \mathrm{G}_i P \right\} ,$$

$$\mathrm{G}_i = \left\{ p_{i1}, p_{i2}, ..., p_{ij} | \mathrm{j} = \mathrm{Num}(\mathrm{G}_i) \, \mathrm{and} \sum_{i=1}^{n} Num(G_i) = \mathrm{m} \right\},$$

$$p_i = \{SC_i, SM_i, SH_i, SN_i\} .$$

G_i stands for the i^{th} group on cloud, and p_i stands for the configure vector of the PM *i*. SC_i, SM_i, SH_i, SN_i respectively stands for the CPU size, the memory size, the disk size and the bandwidth size of P_i.

Definition 2. Let VT stands for the VM type's set. VT is defined as follows.

$$\mathrm{VT} = \{vt_1, vt_2, vt_3, \ldots \ldots, vt_t\}),$$

$$vt_i = \{sc_i, sm_i, sh_i, sn_i).$$

vt_i stands for the configure vector of the VM of type *i*, and sc_i, sm_i, sh_i, sn_i respectively stands for the CPU size, the memory size, the disk size and the bandwidth size of vt_i.

Definition 3. $v_i = \{sc_i, sm_i, sh_i, sn_i)$ can be deployed on $p_i = \{SC_i, SM_i, SH_i, SN_i\}$ only when

$$sc_i \le SC_i \ ; sm_i \le SM_i; sh_i \le SH_i; sn_i \le SN_i.$$

Definition 4. $p_j \le p_i \ (i, j \in (0, m])$

$$\text{s.t. } SC_j \le SC_i \ ; SM_j \le SM_i; SH_j \le SH_i; SN_j \le SN_i$$

Definition 5. Let V be the set of the VMs vectors which are deployed in cloud.

$$\mathrm{V}=\left\{\begin{matrix}V_1,V_2,\ldots\ldots,V_m|V_j=\{v_{j1},v_{j2},\ldots\ldots,v_{jk}\},\\ k\in N\ and\ \ j\in[1,\ \ m]\end{matrix}\right\}$$

Definition 6. V_j represents the set of the VMs on the PM p_j, the v_{ji} represents the i^{th} VM on the PM p_j.

Let W_v be the pre-supposed weighted matrix of the VMs, W_p be the pre-supposed weighted matrix of the PMs. The W_v and W_p are stored in the VM scheduler as follows.

$$\boldsymbol{W}_v=\begin{bmatrix}wv_{11} & wv_{12} & \ldots & wv_{14}\\ wv_{21} & wv_{22} & \ldots & wv_{24}\\ \ldots & \ldots & \ldots & \ldots\\ wv_{t1} & wv_{t2} & \ldots & wv_{t4}\end{bmatrix}$$

$$\boldsymbol{W}_p=\begin{bmatrix}wp_{11} & wp_{12} & \ldots & wp_{14}\\ wp_{21} & wp_{22} & \ldots & wp_{24}\\ \ldots & \ldots & \ldots & \ldots\\ wp_{m1} & wp_{m2} & \ldots & wp_{m4}\end{bmatrix}$$

$$\mathrm{W_{v_s}}=(\mathrm{wv_{s1}},\mathrm{wv_{s2}},\mathrm{wv_{s3}},\mathrm{wv_{s4}})$$

$$\mathrm{W_{p_j}}=(\mathrm{wp_{j1}},\mathrm{wp_{j2}},\mathrm{wp_{j3}},\mathrm{wp_{j4}})$$

$$\mathrm{wv_{s1}}+\mathrm{wv_{s2}}+\mathrm{wv_{s3}}+\mathrm{wv_{s4}}=1$$

$$\mathrm{wp_{j1}}+\mathrm{wp_{j2}}+\mathrm{wp_{j3}}+\mathrm{wp_{j4}}=1$$

$\boldsymbol{W}_{v_s}$ stands for the weighted vector of v_s, while the $\boldsymbol{W}_{p_j}$ stands for the weighted vector of p_j. And $wv_{s1},wv_{s2},wv_{s3},wv_{s4}$ respectively represents the pre-supposed weight of CPU, memory, hard disk, network for the v_s, while $wp_{j1},wp_{j2},wp_{j3},wp_{j4}$ has the similar meaning. So we can use $\boldsymbol{W}_x$ (X can be a VM or a PM) to represent the performance weight vector of X.

4.2 LOAD VALUE MEASUREMENT MODEL

Let $bl(j)$(j∈[1,m]) be the load value of the PM p_j, O as the load balance value of the scheduling domain. Function $f_n(j)$ returns the number of the VMs deployed on the PM p_j, while $H_p(p_j)$ represents the load value of PM p_j, and $H_v(j,i)$ stands for the load value of the VM v_{ji}. The performance weighted strategy calculates the load value according to the following Equations. σ is a systematic parameter which can be set by the system administrator.

$$H(x)=\boldsymbol{W}_x\times\boldsymbol{S}_x, \tag{2}$$

$$bl(j)=\sigma\times H_p(p_j)+(1-\sigma)\times\sum_{i=1}^{f_n(j)}H_v(j,i)/\ f_n(j). \tag{3}$$

$$O=\sqrt{\sum_{j=1}^{m}\left(bl(j)-|bl|\right)^2}/m\ ,$$

$$\sigma\in(0,1)\ ,\qquad\left(|bl|=\sum_{i=1}^{m}bl(i)/m\right). \tag{4}$$

Lemma 1. Based on the definitions above, we can deduce that the smaller the O is, the higher the cloud load balance degree is.

Lemma 2. Based on the configure set of PMs and VMs, we can get the VM number matrix $\boldsymbol{R}$, while the r_{ij} represents the max number that the vt_i can be deployed on the p_j. The detail is as follows:

$$\boldsymbol{R}=\begin{bmatrix}r_{11} & r_{12} & \ldots & r_{1m}\\ r_{21} & r_{22} & \ldots & r_{2m}\\ \ldots & \ldots & \ldots & \ldots\\ r_{t1} & r_{t2} & \ldots & r_{tm}\end{bmatrix}.$$

$$r_{ij}=min\{l_1,\ l_2,\ l_3,\ l_4\}\ (\ i\in[1,\ t],\ j\in[1,\ m]).$$

$$l_1=\ SC_{p_j}/sc_{v_i},l_2=\ SM_{p_j}/sm_{v_i}.$$

$$l_3=\ SH_{p_j}/sh_{v_i}\ ,l_4=\ SN_{p_j}/sn_{v_i}.$$

Lemma 3. When a new VM q_x is requested whose type number is be k, we can find the corresponding weighted vector in W_v. Suppose $H'_v(v_x)$ returns the presupposed load value for q_x, while $H'_p(p_j)$ returns the pre-supposed load value of the PM p_j if the v_x is deployed on it, and $\boldsymbol{S'}_{p_j}$ returns the pre-supposed load data vector of the PM p_j. Then we can deduce the following Equation.

$$\boldsymbol{S'}_{p_j}\approx\boldsymbol{S}_{p_j}+1/r_{kj}\ ,$$

$$H'_p(p_j)=\boldsymbol{W}_{p_j}\times\boldsymbol{S'}_{p_j}\approx H_p(p_j)+\boldsymbol{W}_{p_j}\times100/r_{jk}, \tag{5}$$

$$H'_v(v_x)=\sum_{i=1}^{f_n(j)}H_v(j,i)/f_n(j). \tag{6}$$

Lemma 4. Given $bl'(j)$ returns the pre-supposed load value of the PM p_j. Then we can deduce it as follows basing on Equation (5) and (6).

$$bl'(j)=\sigma\times H'_p(p_j)+$$
$$1-\sigma\cdot\left(\sum_{i=1}^{f_n(j)}H_v(j,i)+H'_v(v_x)\right)/\left(f_n(j)+1\right)\ . \tag{7}$$

Lemma 5. Suppose $\Delta bl(j)$ as the changes in the load values of p_j if the v_x is deployed on itself, then we can deduce as follows.

$$\Delta bl(j)=bl'(j)-bl(j)$$
$$=\sigma\times H_p(p_j)+\sigma\times100\times\overline{W_{p_j}}/r_{jk}+\ . \tag{8}$$
$$(1-\sigma)\times\left(\sum_{i=1}^{f_n(j)}H_v(j,i)+H'_v(v_x)\right)/\left(f_n(j)+1\right)$$

When $f_n(j)$ is very great, we can deduced the below Equation.

$$\Delta bl(j) \approx \sigma W_{p_j} / r_{jk} + (1-\sigma) H'_v(v_x) / f_n(j). \quad (9)$$

Lemma 6. Suppose $\Delta O(j)$ as the changing value in the load balance value of the cloud scheduling domain if the v_x is deployed on itself, then we can deduce as follows.

$$|O'| = \left(\sum_{i=1 \text{ and } i \neq j}^{m} bl(i) + bl'(j) \right) / m, \quad (10)$$

$$\Delta O(j) = \sqrt{\sum_{i=1 \text{ and } i \neq j}^{m} \left(bl(i) - |O'|\right)^2 + \left(bl'(j) - |O'|\right)^2 / m - O}. \quad (11)$$

5 Scheduling algorithm

5.1 ALGORITHM SUMMARY

Suppose there are n VMs which the user layer has requested, while the request set is called Q:

$$Q = \{q_1, q_2, q_3, \ldots\ldots, q_n\} | qi \in VT.$$

Then the main work of the deploying and scheduling for VMs is to find the mapping f:

$$f: Q \xrightarrow[q_i \longrightarrow p_j]{deployed} P \quad (q_i \text{ is deployed on } p_j).$$

The details are as follows.

Adding all the alarm tasks into the alarm queue Q_R, and invoking the alert processing module to process each PM in queue sequentially.

Adding the user-specified optimization tasks into the optimization queue Q_O, and calling the optimization module to optimize each PM in queue.

Adding new requested task into the new task queue Q_N, and calling the VM deployment scheduling module to treat each request in order.

5.2 PREREQUISITES AND RESTRICTIONS

Never deny the user's request ($v = \{sc, sm, sh, sn\}$) except when:

$c \leq \sum_{i=1}^{m} SC_i$ or $sm \leq \sum_{i=1}^{m} SM_i$

or $sh \leq \sum_{i=1}^{m} SH_i$ or $sn \leq \sum_{i=1}^{m} SN_i$.

Only allow the user to request one of the VM types among the given VT.

Each PM and VM can be monitored, and when there was a PM or VM load had exceeded the specified threshold, the monitor would alarm.

The first group G_1 is the default initial scheduling domain on the cloud.

5.3 THE DETAILS OF THE ALGORITHM

5.3.1The main process

Input:

$P = \{G_i \mid i \in (0, n) \text{ and } G_i P$

$P = \{p_1, p_2, \ldots, p_m \mid m \in N\}$

1) V, W_v, R, W_p;
2) $S = \{S_x \mid x \in P \text{ or } x \in V\}$;
3) The queues Q_R, Q_O, Q_N.

Output: $f: Q_N \xrightarrow[q_i \longrightarrow p_j]{deployed} P$; finish status.

Steps

```
1)  while(QR!=NULL)
    {
    q=QR->head;
     if (dynamicsheduler (q->id, q->threshold, (P-QR))= = SUCCESS)//
process PM q
     {QR->head=q->head; delete (q);}
     else
     go to 4);
     }
2)  while(QO !=NULL)
    {     q=Qo->head;
      if (dynamicsheduler (q->id, S, (P-Qo))= =SUCCESS) // process
      PM q
        {Qo->head=q->head;
         delete(q);
         }
      else
      go to 4);
    }
3)  while(QN !=NULL)
    {q= QN->head;
     if (staticsheduler (q->id, P)= =0)
        // process PM q
       go to 4);
     else {QN->head=q->head;
        new p;
        p->id= staticsheduler(q-> id, P);
        insert(q ⟶ p) to f;
        Update(V, q ⟶ p);
        deploy(q,p);//deploy q on p
        delete(q);
        }
    }
4)  if (Num(the current Gi)= =m)
    //if all the PMs in the current scheduler domain
         return FAIL;
    else {
    start Gi+1;merge(Gi, Gi+1);
    }
```

5.3.2 The VM placement module

staticscheduler

Input: the requested new VM q; ()

Output: the id of the target PM (0 stands for failure)

Steps:

```
1)  for(i=1;i<=num(G);i++)
    {p=G(i);
     if(S(p) ≈0) return i;
    }
```

2) for(i=1;i<=num(G);i++)
if (G(i)>q) insert (G(i),Q_L);
{if(Q_L!=NULL)
{Scan Q_L orderly;
if (m> system threshold number)
//if m is very large
return the first j in $\{j|j \in \min_{j\in[1,num(Q_L)]} \Delta bl(j)\}$;
else
return the first j in $\{j|j \in \min_{j\in[1,num(Q_L)]} \Delta O(j)\}$;
}
else
return dynamicsheduler (q->id, the specific threshold, G);
}

5.3.3 The dynamic migration scheduling module

dynamicsheduler

Input: ID (PM id or new VM type number);
The default threshold of PM load; the current schedule domain G;
Output: FAIL or SUCCESS

Steps:

1) Make *P* to *P'*(*P'* is in descending order according to the available size of resources);
2) if the module is invoked by Q_O or Q_R
{delete p whose id = =ID from *P'*;
v=Max (V_{id}); //v is the max VM on p;
}
else v=vt_{id}; //v is the requested VM;
3) for(x=1;x<num(*P'*),x++)
{
if (there is *V'* on V_x which meet the following condition:
a); (V_x- *V'*)>*v*
b) for each $v' \in V'$
staticsheduler (*v'*->id, P')! =0
)
{
for each $v' \in V'$ deploy (*v'*, *P'*);
return SUCCESS;
}
else return FAIL;
}

6 Experimental result and analysis

This section shows that the SAST is feasible and exact. This experiment is simulated with Java programming language, using a set of simulated experiment set to compare the performance of the round robin algorithm, the best fit heuristic packing algorithm and the SAST.

6.1 CONFIGURATION

1. The configuration of the PMs and VMs is described in Table 2. The PMs consists of 10 nodes and 2 kinds of configuration. There are 6 PMs about node1, and 4 PMs about node 2; The VMs has two types, one is the high computing (HC), another is the high storage (HS). Each type is divided into big, medium and small class, so there are six different VM configurations. Clusters are divided into 10 groups, each group of only one PM. Each CPU has a core (Intel(R) Xeon (R) E5606). There are 10 PMs with two types of configuration.

TABLE 2 Experiment configuration

Type	CPU (G)	Memory (G)	Storage (GB)	Network (MB)
node1	16	96	20*1024	1000
node2	16	96	10*1024	1000
VM_A	8	8	200	100
VM_B	8	4	100	100
VM_C	8	2	50	100
VM_D	4	8	200	100
VM_E	4	4	100	100
VM_F	4	2	50	100

2. All strategies using the same set of monitoring data.

3. In each strategy the VMs are requested at random for 15 times.

4. When talking about round robin strategy, let's suppose that Wp=(0.25,0.25,0.25,0.25), and for SAST, σ=0.6,and the setting of Wv is shown as Table 3.

TABLE 3 W for six VMs

w	VM_A	VM_B	VM_C	VM_D	VM_E	VM_F
w_1	0.3	0.2	0.4	0.2	0.5	0.2
w_2	0.25	0.25	0.2	0.2	0.15	0.15
w_3	0.2	0.3	0.2	0.4	0.2	0.5
w_4	0.25	0.25	0.2	0.2	0.15	0.15

6.2 RESULTS AND ANALYSIS

The number of PMs started of the SAST is the same with the best fit heuristic packing algorithm at each application as shown in Figure 2, and the early number were less than the largest PM number. But when the system entered into a stable state, the load balance degree of cloud of the SAST is relatively high, particularly as shown in Figure 3. The abscissa represents the number of VMs, and the ordinate represents the value of load balance value of the cloud.

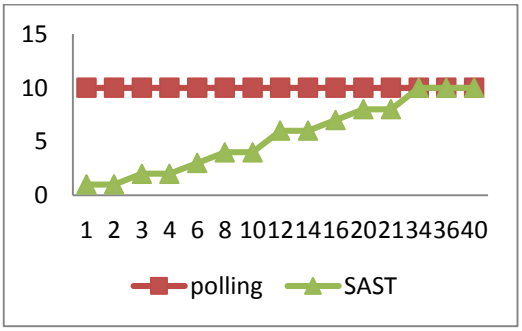

FIGURE 2 The PMs started

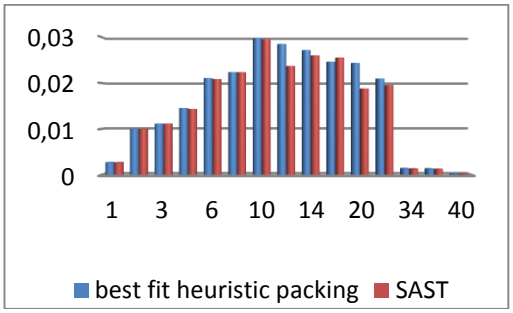

FIGURE 3 The load balance value

We let two strategies applied 20 VMs at random for 4 times, the total number of migration at the steady state of the cloud platform was compared and shown in Table 4.

TABLE 4 The total number of migration

Strategy	Random Equilibrium	SAST
the number of test 1	4	2
the number of test 2	4	3
the number of test 3	3	0
the number of test 4	5	2

7 Conclusions

Considering the user requests of virtual resources and setting the weight of performance parameters to calculate the load balance value of the cloud, we present a self-aware strategy for VMs placement on clouds. It achieved better load balance of cloud, and at the same time, it minimized the number of the started PMs on clouds to reduce energy consumption. However, the time complexity of the SAST is relatively high when it is compared with the round robin algorithm and other traditional algorithms. We will emphasize with the security on cloud in the future work.

Acknowledgments

This work is supported by the national science of nature fund project: The Computing Plat tern based on Formal Area (x2jsb55101680) and the Guangdong province science of nature fund research group project: Grid Theory Model and Analysis (x2jsb6110010).

References

[1] The NIST Definition of Cloud Computing *National Institute of Standards and Technology* 2011
[2] FORREST W 2008 How to cut data centre carbon emissions http://www.computerweekly.com/feature/How-to-cut-data-centre-carbon-emissions
[3] Goldberg R P 1974 *IEEE Computer* **7**(6) 34-45
[4] Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A 2003 Xen and the art of virtualization *ACM SIGOPS Operating Systems Review* **37** 164-77
[5] Tian W, Zhao Y 2011 Cloud Computing Resource Scheduling Management *National defense science and technology university press*
[6] Hai J 2009 Computing System Virtualization - the Principle and Application *Tsinghua university press* (*in Chinese*)
[7] Verma A, Ahuja P, Neogi A 2008 Mapper: power and migration cost aware application placement in virtualized systems *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware* 243-64
[8] Bobroff N, Kochut A, Beaty K 2007 Dynamic placement of virtual machines for managing sla violations *Integrated Network Management IM'07 &10th IFIP/IEEE International Symposium* 119-28
[9] Cardosa M, Korupolu M R, Singh A 2009 Shares and utilities based power consolidation in virtualized server environments *Integrated Network Management IM'09 & IFIP/IEEE International Symposium Papers* **31** 327-34
[10] Tan T X, Cameron K 2009 An Assessment of Eucalyptus Version 1.4 2009-929-07 Calgary *Department of Computer Science University of Calgary*
[11] Wikipedia. Eucalyptus. http://en.wikipedia.org/wiki/Eucalyptus
[12] Corradi A, Fanelli M, Foschini L VM consolidation: A real case based on OpenStack Cloud *Dipartimento di Elettronica, Informatica e Sistemistica (DEIS) University of Bologna Italy*
[13] Opennebula 2012 *Opennebula Scheduling Policies 2.0* http:// www.opennebula.org
[14] Aiash M, Mapp G, Gemikonakli O 2014 Secure live virtual machines migration: issues and solutions *Advanced Information Networking and Applications Workshops (WAINA) 28th International Conference* 160-5
[15] Bobroff N, Kochut A, Beaty K 2007 Dynamic Placement of VMs for Managing SLA Violations *Proceedings of 10th IFIP/IEEE International Symposium on Integrated Network Management papers* **118** 119-28
[16] Singh A, Korupolu M, Mohapatra D 2008 Server-Storage Virtualization: Integration and LoadBalancing in Data Centers *Proceeding of the 2008 ACM/IEEE conference on Supercomputing (SC'08) papers* **49** 1-12
[17] Tsakalozos K, Roussopoulos M, Delis A 2011 VM: placement in non-Homogeneous Iaas-clouds *Proceedings of the 9th international conference on Service-Oriented Computing* 172-87
[18] Breitgand D, Epstein A 2011 SLA-aware placement of multi-VM elastic services in compute clouds *Proceeding of IFIP/IEEE International Symposium on Integrated Network Management* 161-8
[19] Zhuang Wei, Gui Xiaolin, Lin Jiancai1, Wang Gang, Dai Min 2013 Deployment and scheduling of vms in cloud computing: an "AHP" approach *Journal Of Xi'An Jiaotong University* **47**(2) y1-y7
[20] Bessis N, Sotiriadis S, Xhafa F, Asimakopoulou E 2013 Cloud scheduling optimization: a reactive model to enable dynamic deployment of virtual machines instantiations *Journal of High Speed Network* **24**(3) 357-80
[21] Yang X, Ma Z, Sun L 2012 Performance vector-based algorithm for VM deployment in infrastructure clouds *Journal of Computer Applications* **32**(1) 16-9
[22] Zhao H, Song B, Shao Y 2012 High-effect resource management strategy in cloud computing environment *Computer Science* **39**(2) 212-5

Authors

Fen Guo, China

Present position, grades: teacher in South China University of Technology, China.
University study: Doctoral students in School of Computer Science & Engineering, South China University of Technology, Guangzhou, China, 2009-present.
Research activities: cloud computing, database, pattern recognition, artificial intelligence.

Huaqing Min, China

Present position, grades: professor in South China University of Technology, China.
University study: PhD in School of Computer Science & Engineering, Huazhong University of Science and Technology, Wuhan, China, in 1998.
Research activities: artificial intelligence, cloud computing, database, pattern recognition.

Ming Yin, China

Present position, grades assistant professor with the School of automation in Guangdong University of Technology, China.
University study: PhD in information & communication engineering from Huazhong University of Science and Technology (HUST), Wuhan, China, in 2006.
Research activities: image/videocoding, image deblurring, sparse representation, unsupervised/semi-supervised data cluster/classification.