Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 829614, 8 pages
http://dx.doi.org/10.1155/2014/829614

Research Article

Global Detection of Live Virtual Machine Migration Based on Cellular Neural Networks

Kang Xie,[1] Yixian Yang,[1,2,3] Ling Zhang,[2] Maohua Jing,[3] Yang Xin,[2] and Zhongxian Li[4]

[1] *College of Information Science and Engineering, Shandong University, Jinan 250100, China*
[2] *Information Security Center, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[3] *Northeastern University & College of Information Science and Engineering, Shenyang 110819, China*
[4] *National Cybernet Security Ltd., Beijing 100088, China*

Correspondence should be addressed to Kang Xie; xiekang1987@sina.com

Received 12 February 2014; Accepted 31 March 2014; Published 6 May 2014

Academic Editor: Antonio Puliafito

In order to meet the demands of operation monitoring of large scale, autoscaling, and heterogeneous virtual resources in the existing cloud computing, a new method of live virtual machine (VM) migration detection algorithm based on the cellular neural networks (CNNs), is presented. Through analyzing the detection process, the parameter relationship of CNN is mapped as an optimization problem, in which improved particle swarm optimization algorithm based on bubble sort is used to solve the problem. Experimental results demonstrate that the proposed method can display the VM migration processing intuitively. Compared with the best fit heuristic algorithm, this approach reduces the processing time, and emerging evidence has indicated that this new approach is affordable to parallelism and analog very large scale integration (VLSI) implementation allowing the VM migration detection to be performed better.

1. Introduction

Cloud computing provides ability to dynamically scale or shrink the provisioned resources as per the dynamic requirements [1]. It is elastic in nature and works on pay as its clients use model. Virtualization as a key concept of cloud computing gives an abstract view of hardware by means of multiple virtual guest operating systems which are known as Virtual machines (VMs) and are instanced on a single physical machine (PM). That is, depending on the capacity of the PM, many VMs can be created on it. It is also clear that the proper management of VMs can improve the resource utilization efficiently and carry out the isolation to multitenancy. Virtual machine monitor (VMM), which is also known as hypervisor, is simply a software and can be used as an interface between PM and VMs. One of the widely used VMM is Xen which allows various VMs to share common hardware in a safe and resource managed condition but without sacrificing the performance [2].

VM migration is a key enabler for dynamic resource management in cloud-based systems. Live VM migration as an extremely powerful tool for the cloud managers transfers state of a VM from one PM to another and can mitigate overload conditions and enables uninterrupted maintenance activities. In the migration process, complete state of running VM, which includes the permanent storage (i.e., the disk image), volatile storage (the memory), the state of connected devices (such as network interface card), and the internal state of the virtual CPU (VCPU), has to be transferred [2]. In this case, VM locations are varied dynamically and the internal state of the VCPU and connected devices are a few kilobytes of data and can be easily sent to the VMM and the target PM. For these reasons, it is clear that VMs can be migrated from overloaded PM to underloaded PM, but it will be only helpful when the efficient live migration techniques are used. Thus, we need an approach to trace and detect the processes migration in VM and mapping relationship between VCPUs and PM from the hypervisor's point of view, in order to verify whether the new scheduler is effective and the result is consistent with the initial idea.

Therefore, to meet the demands of processing time-saving, a lightweight algorithm of live VM migration

detection (VMMD) based on cellular neural network (CNN) is proposed in this paper. CNN has features with multidimensional array of neurons and realizable paradigm of parallel computation; the processing time is unrelated with the data dimension. Moreover, it can be implemented by very large scale integration (VLSI), which makes neural networks easily implemented [3]. Therefore, it has been widely used in many fields [3–7], such as the classification and recognition of moving targets [4, 5], path detection of mazes [6], and microarray image reconstruction [7]. To the best of our knowledge, few reports have been published on the application in live VM migration detection.

The rest of this paper is organized as follows. Section 2 provides an overview of the most adopted VM migration techniques. In Section 3, the proposed algorithm of live VMMD based on CNN is discussed. Section 4 explains the details for designing CNN parameters via bubble sort particle swarm optimization (BSPSO) algorithm. After that, Section 5 presents the experimental results, followed by the conclusions in Section 6.

2. Related Works

In the proposed VM migration policy, VM migration is triggered when the data transfer time crosses a certain threshold due to the unstable network. Here, the threshold can be determined by a time-related service level agreement (SLA) between the cloud facility provider and the cloud user [8]. To solve the problem of host overload detection, a novel approach based on a Markov chain model is proposed by maximizing the mean intermigration time under the specified quality of service (QoS) goal [9]. In [10, 11], migration strategies are generally based on the CPU utilization, and when CPU utilization of a host exceeds a certain threshold, tasks implementing on the VM with the highest CPU utilization on the host will be migrated out. These strategies are generally lack of resources sensitivity and can not realize the dynamic resource allocation according to the actual conditions of the load.

In [12], Celesti et al. propose a composed image cloning (CIC) methodology able to reduce the cost of the VM disk-image relocation over a WAN. In [13], a new live VM migration strategy is proposed by using the load characteristics to implement hotspots detection, selection of VM, and migration destination host according to some multithreshold patterns. Srikantaiah models the migration problem as a modified bin packing problem in [14]. Firstly, the optimal point is determined from profiling data. Then, the heuristic algorithm is used to maximize the sum of the Euclidean distances of the current allocations to the optimal point at each server. Luo et al. [15] propose an advanced VM dynamic consolidation system through live migration approach. The system consists of three main components: load monitor which collects resources usage statistics from each server node, relocation planner which calculates the relocation plan by analyzing the resource usage histories, and VM controller which activates live migration to server nodes according to the results from relocation planner. In [16], Li et al. propose a live migration strategy for VM, which combines with performance predicting algorithm. According to the average utilization of the CPU, memory, I/O, and network bandwidth, the proposed strategy makes a serial of judgments about migration, such as whether a migration is triggered, what VM should be migrated, and where is the destination PM of the VM. But these existing researches are generally heuristic based or heavily rely on statistical analysis of historical data; thus, the optimization procedure has to be conducted after a relatively long period to obtain statistics.

To trace the migration process in VM, Zhang et al. [17] present a demo process migration tracing engine for monitoring the migration of process on VCPUs and VCPUs on the cores of physical processor based on Linux 2.6 and Xen 3.2. In [18], Liu et al. design a novel approach CR/TR-Motion that adopts recovery and trace technology to provide fast, transparent VM migration. With execution trace logged on the source PM, a synchronization algorithm is performed to orchestrate the running source and target VM until they get a consistent state. According to the above issue, it is clear that it is important for the proper management of virtualization resources information and security events, monitoring and analysis of VM situations, and migration policies. Therefore, we study the tracing method for history of infected VMs following the analysis of the VM migration lifecycle status and running states.

3. The Processing of VMMD Based on CNN

CNN is a large scale nonlinear locally connected analog circuit which processes signal in real time that was first proposed by the Professor Chua and Dr. Yang who came from the University of California in 1988 [3]. A CNN is composed of basic processing units called cells. Each cell, denoted by $c(i, j)$, is connected to its neighboring ones; therefore, only the adjacent cells can directly connect with each other, others interaction are through dynamic continuous propagation effects. Each cell has the same structure, composed by a small amount of linear and nonlinear circuit elements.

The circuit equations of a cell which satisfy the KCL and KVL are easily derived as follows.

State equation is

$$C\frac{dv_{xij}(t)}{dt} = -\frac{1}{R}v_{xij}(t) + \sum_{c(k,l)\in N_r(i,j)} A(i,j;k,l)\, v_{ykl}(t) + \sum_{c(k,l)\in N_r(i,j)} B(i,j;k,l)\, v_{ukl}(t) + I. \quad (1)$$

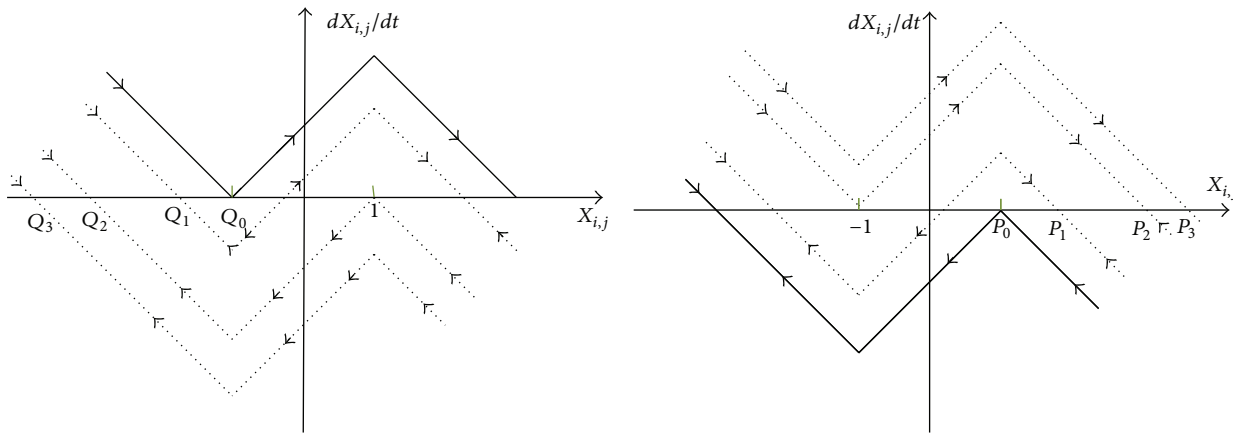
Output equation is

$$v_{ykl}(t) = f\left(v_{xij}\right) = \frac{1}{2}\left(\left|v_{xij}(t)+1\right| - \left|v_{xij}(t)-1\right|\right). \quad (2)$$

Constraint conditions are

$$\left|v_{xij}(0)\right| \le 1, \qquad \left|v_{uij}(0)\right| \le 1, \quad (3)$$

where $1 \le i \le M$, $1 \le j \le N$. The dynamics of a CNN has both output feedback and input control mechanisms. The



(a) The trajectory of Local Rule 1

(b) The trajectory of Local Rule 2

Figure 1: The dynamic trajectory of cell for different local rules.

output feedback effect depends on the interactive parameter $A(i, j; k, l)$ and the input control effect depends on $B(i, j; k, l)$. The key of CNN-based solution is to design the appropriate parameters of the feedback matrix A and control matrix B which represent the connection weight between cells.

Virtualization provides virtualized view of resources used to instantiate VMs. Once a VM is instantiated, a resource monitoring engine which is called VMM or hypervisor will track the resource usage and performance indicators related to the applications of the VM; it also manages and multiplexes access to the physical resources, maintaining isolation between VMs at all times.

Let $\mathrm{VM}_n = vm_n^1 \cup vm_n^2 \cup \cdots vm_n^m$ denote the static state matrix of VM numbered n, composed of the union of m continue time state components vm_n^1, vm_n^2, ..., and vm_n^m. The allocation matrix of VM $P_g = \{P_{11}, \ldots, P_{i,j}, \ldots, P_{l,s}\}$ can be divided into the state set of VM $V = \{\mathrm{VM}_1, \ldots, \mathrm{VM}_l\}$ and physical host (PH) $H = \{H_1, \ldots, H_s\}$, if a VM numbered v is allocated into the host named h, $P_{hv} = 1$; otherwise, $P_{hv} = -1$. Any allocate solutions must meet the capacity constraints of PHs as follows:

$$\begin{aligned} \forall h \in \{1, \ldots, n\} \quad & \sum_{v=1}^{l} P_{hv} \times \mathrm{CPU}\left(V_v\right) \le \mathrm{CPU}\left(H_h\right), \\ \forall h \in \{1, \ldots, n\} \quad & \sum_{v=1}^{l} P_{hv} \times \mathrm{Mem}\left(V_v\right) \le \mathrm{Mem}\left(H_h\right), \end{aligned} \tag{4}$$

where the idle host will be closed to save energy.

Let VM_0 denote the VMM which is called triggering cells, and let $\mathrm{VM}' = \mathrm{VM} \setminus \mathrm{VM}_0$ as the initial state denote the remainder pattern, to be deleted from VM. VM' is applied as input to the global state detection of VM by the cloning templates given by (5) [3]. Since $\mathrm{VM}' \in \mathrm{VM}$, each state of VM at the initial time $t = 0$ can assume the three combinations of (input, initial state) $= (u_{i,j}, x_{i,j}(0))$; namely, $(\mathrm{VM}_{\mathrm{suspend}}, \mathrm{VM}_{\mathrm{suspend}}) = (-1, -1)$, $(\mathrm{VM}_{\mathrm{copy}}, \mathrm{VM}_{\mathrm{suspend}}) = (1, -1)$, $(\mathrm{VM}_{\mathrm{copy}}, \mathrm{VM}_{\mathrm{copy}}) = (1, 1)$. Consider

$$A = \begin{bmatrix} 0 & \frac{c}{2} & 0 \\ \frac{c}{2} & a & \frac{c}{2} \\ 0 & \frac{c}{2} & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & -\frac{c}{2} & 0 \\ -\frac{c}{2} & b & -\frac{c}{2} \\ 0 & -\frac{c}{2} & 0 \end{bmatrix}, \qquad Z = z, \tag{5}$$

where a, b, c, and z are real numbers and $c > 0$.

The steady state output $y_{i,j}(\infty)$ of each cell $P_{i,j}$ belonging to any connected component obeys two static local rules and one dynamic global rule.

Local Rule 1. If $z \le a + b - 1$, $(u_{i,j}, x_{i,j}(0)) = (-1, -1)$, thus $y_{i,j}(\infty) = -1$, independent of $(u_{k,r}, x_{k,r}(0))$ of its neighbors states.

Local Rule 2. If $z \ge 1 - a - b$, $(u_{i,j}, x_{i,j}(0)) = (1, -1)$, thus $y_{i,j}(\infty) = 1$, independent of $(u_{k,r}, x_{k,r}(0))$ of its neighbors states.

Proof. Consider

$$\begin{aligned} x_{i,j}^{\bullet}(t) &= -x_{i,j}(t) + a y_{i,j}(t) + b u_{i,j} \\ &\quad + \frac{c}{2} \sum_{(k,r) \in I} \left[y_{i+k,j+r}(t) - u_{i+k,j+r}(t) \right] + z \\ &= -x_{i,j}(t) + a y_{i,j}(t) + w_{i,j}(t), \end{aligned} \tag{6}$$

where $i = 1, 2, \ldots, l$, $j = 1, 2, \ldots, s$.

Assume that the parameters satisfy $a > 1/R_x = 1$, and then each cell of the CNN must settle at a stable equilibrium point after the transient has decayed to zero, and $\lim_{t \to \infty} v_{yij}(t) = \pm 1$, $1 \le i \le M$, $1 \le j \le N$ guarantee that our CNNs have binary-value outputs. This property is very important for detection problems in live VM migration.

(i) If $(u_{i,j}, x_{i,j}(0)) = (-1, -1)$, in order to guarantee Local Rule 1 to be hold, $y_{i,j}(\infty) = -1$. From Figure 1(a), $x_{i,j}^{\bullet} \le 0$,

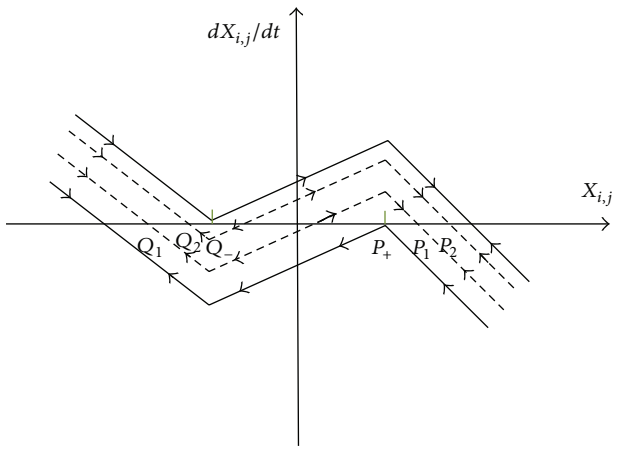


Figure 2: The dynamic trajectory of cell for global rule.

the dynamic trajectory of cell $P_{i,j}$ must tend to an equilibrium $Q_0 = -1$, and the following inequality must be satisfied.

Thus,

$$\begin{aligned} 1 - a + w_{i,j}(t) &\le 0, \\ z &\le a + b - 1. \end{aligned} \tag{7}$$

(ii) If $(u_{i,j}, x_{i,j}(0)) = (1, -1)$, in order to guarantee Local Rule 2 to be hold, $y_{i,j}(\infty) = 1$. From Figure 1(b), $\dot{x}_{i,j} \ge 0$, and the dynamic route of cell $P_{i,j}$ is not below the curve with $w_{i,j} = 1 - a$ and finally tends to an equilibrium $P_0 = 1$; the following inequality must be satisfied:

$$a - 1 + w_{i,j}(t) \ge 0. \tag{8}$$

Thus,

$$z \ge 1 - a - b. \tag{9}$$

□

Global Rule. If $z > -a - b - 2c - 1$, $(u_{i,j}, x_{i,j}(0)) = (1, 1)$, and there exists an adjacent triggering cell or a directional connected path defined by I from cell $P_{i,j}$ to some marked cell P_{i^*,j^*}, $y_{i,j}(\infty) = 1$. Otherwise, $z < 1 - a - b + 2c$, $y_{i,j}(\infty) = -1$.

Proof. If there exists a directional connected path from cell $P_{i,j}$ to some marked cell P_{i^*,j^*}, then there exists some connected path from cell $P_{i,j}$ to some cell $P_{i',j'}$ and there is at least one adjacent triggering cell $P_{i',j'}$ connected to cell P_{i^*,j^*}, $(u_{i,j}, x_{i,j}(0)) = (1, 1)$ and $y_{i,j}(\infty) = -1$. In this case, $\dot{x}_{i,j} > 0$.

Thus,

$$1 + a + w_{i,j}(t) > 0, \tag{10}$$

$$z > -a - b - 2c - 1. \tag{11}$$

If there neither exit an adjacent triggering cell or a directional connected path defined by I from cell $P_{i,j}$ to some marked cell P_{i^*,j^*}, $(u_{i,j}, x_{i,j}(0)) = (1, 1)$. The dynamic trajectory of cell must tend to an equilibrium P_+ indicated in Figure 2; that is, $\dot{x}_{i,j} < 0$.

Hence,

$$1 + a + w_{i,j}(t) < 0, \tag{12}$$

$$z < -1 - a - b + 2c. \tag{13}$$

If (11) and (13) are satisfied, then the global rule holds. In summary, we complete the proof of Local Rules 1 and 2 and global rule. □

4. The Design of CNN Template Parameters Based on BSPSO Algorithm

CNN is a typical nonlinear dynamic system with ability of optimization. The Lyapunov function, $E(t)$, of a CNN by the scalar function is defined as follows:

$$\begin{aligned} E(t) = &-\frac{1}{2}\sum_{(i,j)}\sum_{(k,l)} A(i,j;k,l)\, v_{yij}(t)\, v_{ykl}(t) \\ &+ \frac{1}{2R_x}\sum_{(i,j)} v_{yij}(t)^2 - \sum_{(i,j)}\sum_{(k,l)} B(i,j;k,l)\, v_{yij}(t)\, v_{ukl} \\ &- \sum_{(i,j)} Z v_{yij}(t), \end{aligned} \tag{14}$$

where $1 \le i, k \le M$; $1 \le j, l \le N$. And it is a monotone decreasing function, always converges to a local minimum, where the CNN produces the desired output. Thus, the detected problem of live VM migration based on CNN can be changed into constrained optimization problem (COP) as follows:

$$\begin{aligned} \min \quad \frac{\partial E_{\mathrm{CNN}}(t)}{\partial t} &= -x_{i,j}(t) + \sum_{k,l \in N_{i,j}(r)} A_{k,l} y_{k,l}(t) \\ &\quad + \sum_{k,l \in N_{i,j}(r)} B_{k,l} u_{k,l} + Z_{i,j}, \end{aligned} \tag{15}$$

$$\text{st} \quad \begin{cases} z \le a + b - 1 \\ z \ge 1 - a - b \\ z > -a - b - 2c - 1 \\ z < -1 - a - b + 2c. \end{cases} \tag{16}$$

Most existing algorithms for COP use the penalty function method to handle constrains, which depends strongly on the penalty parameter [19, 20]. PSO as a global search optimization algorithm has been widely used in COP [21]. The algorithm was inspired by the social behavior of a flock of birds when searching for food. The potential solutions are denoted as particles, fly in the search space exploring for better regions. In PSO, each particle has a current position and a velocity and the particle finds the best solution through exchanges information with other experienced particles.

Assume that a swarm $P(k)$ is composed of N particles in the D-dimensional solution space, where the position vector of particle i is $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ and $V_i = (v_{i1}, v_{i2}, \ldots, v_{id})$ denotes the velocity vector. $P_i = (p_{i1}, p_{i2}, \ldots, p_{id})$ is the

optimal solution vector experienced, that is, the best fitness solution p_{best} found by particle i. In other words, $P_g = (p_{g1}, p_{g2}, \ldots, p_{gd})$ denotes the best fitness solution g_{best} found by all of particles. The velocity and position update equations of each particle are shown as follows:

$$V_i(t+1) = \omega * V_i(t) + c_1 * r_1 * \left(p_{\text{best}} - V_i(t)\right) + c_2 * r_2 * \left(g_{\text{best}} - V_i(t)\right), \quad (17)$$
$$X_i(t+1) = X_i(t) + V_i(t+1),$$

where $i = 1, 2, \ldots, d$. The inertia weight ω is a factor used to control the balance of the search algorithm exploitation. The acceleration coefficient c_1 moderates the maximum step size toward the global best particle, while another coefficient c_2 moderates the step size toward the best personal position of that particle. They are bounded by $0 < c_1, c_2 < 2$. r_1 and r_2 are random value in the range of $[0, 1]$ and usually selected its range artificially to reduce the probability that $V_i(t)$ and $X_i(t)$ escape from the search space.

There are no selection, crossover, and mutation operations to train the parameters of neural network by using PSO. The algorithm is simple, quick convergence, and the training precision is high, but, when a particle finds the local optimal solution, it will stop searching in the solution space, while other particles will quickly move closer to this particle; therefore, the algorithm is easy to fall into premature convergence. In addition, when the algorithm gets into a local optimum, an infeasible solution replacing mechanism is given to improve the search capability in this paper.

The basic idea of the proposed algorithm was to put feasible solutions and infeasible solutions into two different containers, respectively. Let $G_{\text{fs}} = \{x_1, x_2, \ldots, x_{n1}\}$ denote the set of feasible solution, and the set of infeasible solution is $G_{\text{ifs}} = \{y_1, y_2, \ldots, y_{n2}\}$, where $n1 + n2 = N$. $F_{\text{fs}}(x)$ and $F_{\text{ifs}}(y)$ are the fitness function of G_{fs} and G_{ifs}, respectively. Considering that the global optimal solutions often locate on or near the boundary of the feasible region for many COPs, we choose some better solutions in G_{ifs} based on bubble sort algorithm (BS) and add to G_{fs} in order to improve the searching ability. According to this, the value of $F_{\text{fs}}(x)$ should be sorted from small to large based on BS algorithm; that is, after BS processing, in the ordered set of feasible solution $G'_{\text{fs}} = \{x'_1, x'_2, \ldots, x'_{n1}\}$, x'_1 is the best fitness solution. Similarly, y'_1 is the best fitness solution in G'_{ifs}.

In the sort processing, the competitive choice follows these rules:

(1) any feasible solution is better than the infeasible solution;

(2) in two feasible solutions, it gets ahead whose value of $F_{\text{fs}}(x)$ is superior;

(3) in two infeasible solutions, smaller degree of $F_{\text{ifs}}(y)$ is superior.

The processing of improved PSO based on BS algorithm is given as follows.

Step 1. In accordance with the restraint condition, the particles' velocity and position are to be initialized. Iteration time (IT) IT = 1.

Step 2. For each particle, calculate the current value of the fitness function $F(x)$ from the formula (15).

Step 3. Compare $F(x)$ which is current experienced with the best fitness function p_{best}, and if current $F(x)$ is smaller than $F(p_{\text{best}})$, the particle will be set to G_{fs} and become the new p_{best}. Otherwise, the particle will be set to G_{ifs}. Then, IT = IT + 1.

Step 4. Update the velocity and position of the particles according to formulas (5) and (6).

Step 5. After n times of iteration, obtain some superior particles in G'_{ifs} according to $F_{\text{ifs}}(y)$ and BS algorithm.

Step 6. Put the choosing particles into G_{fs} and get the best fitness function g_{best} of current globally experienced in G_{fs} based on BS algorithm.

Step 7. Determine whether the algorithm met the rule of iteration times, if it satisfied the condition then go to Step 8; otherwise, return to Step 2.

Step 8. Output the value of parameters a, b, c, and z.

The flow diagram of BSPSO is as Figure 3.

5. The Simulation Results and Analysis

In this Section, we first introduce the performances of our BSPSO algorithm for the design of CNN template parameters, including the ability to get the template solution and the comparison results with GA, PSO BSPSO, and sort algorithm. Then, the performance of our VMMD model based on CNN algorithm is evaluated, such as the effects of VMMD model on migration lifecycle and the comparison results with other published existing algorithms.

5.1. The Experimentation Results of BSPSO Algorithm. In our study, we use MATLAB 7.6.0 software on the PC of 2 G memory to carry out this simulation experiments. In order to ensure stability and convergence of the BSPSO algorithm, accelerating factors c_1 and c_2 are made to 1.49, inertia factor is equal to 0.729, r_1 and r_2 are the random number in the interval $[0, 1]$, the initial state $v_{xij}(0)$ and $v_{yij}(0)$ of the randomly selected center cells are set to −1, other parameters $C = 10^{-9}F$, $R_x = 1\,\Omega$, and the maximum iterations is 2000 times. Finally, the algorithm gets the template solution as follows:

$$A = \begin{bmatrix} 0 & 0.64 & 0 \\ 0.64 & 6.32 & 0.64 \\ 0 & 0.64 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & -0.64 & 0 \\ -0.64 & 2.19 & -0.64 \\ 0 & -0.64 & 0 \end{bmatrix}, \quad Z = -7.36. \quad (18)$$

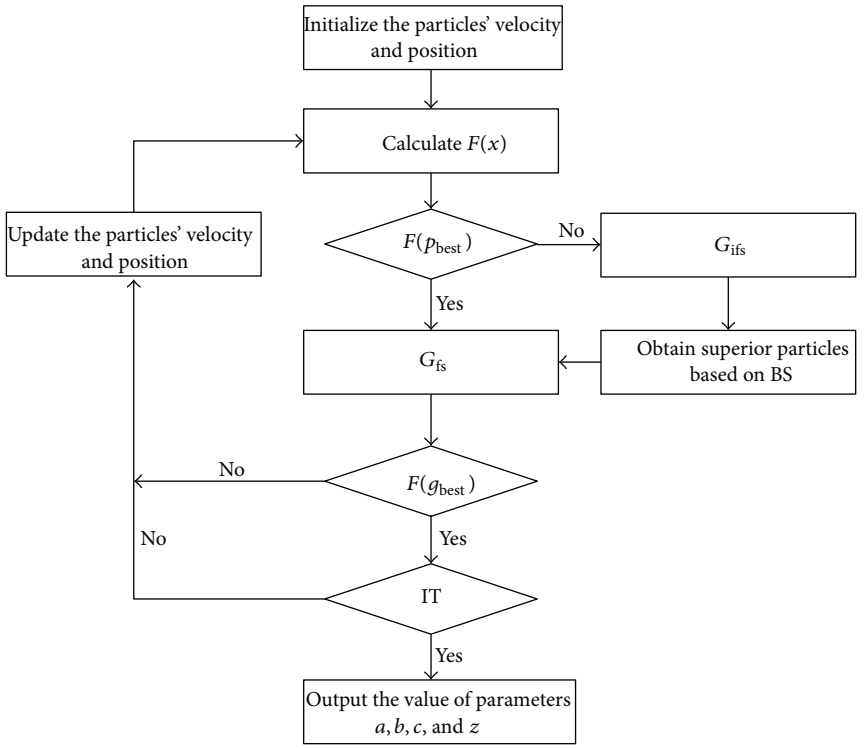


Figure 3: Flow diagram of BSPSO.

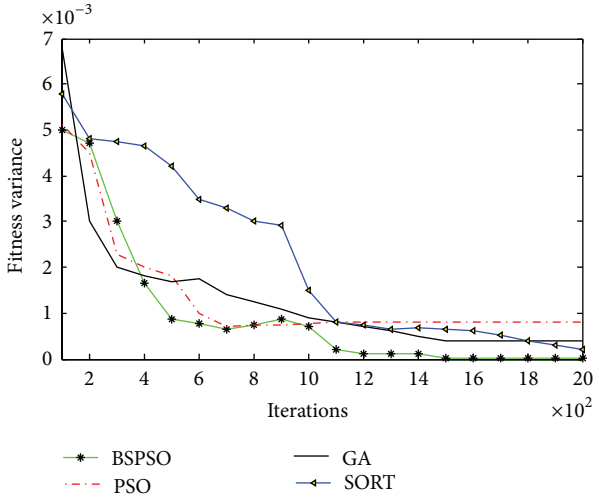


Figure 4: Simulation result of the optimal solution efficiency.

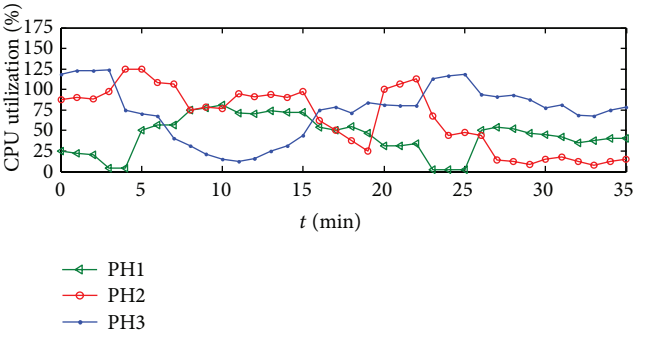


Figure 5: The sum of CPU percentage usage of VMs on each PH.

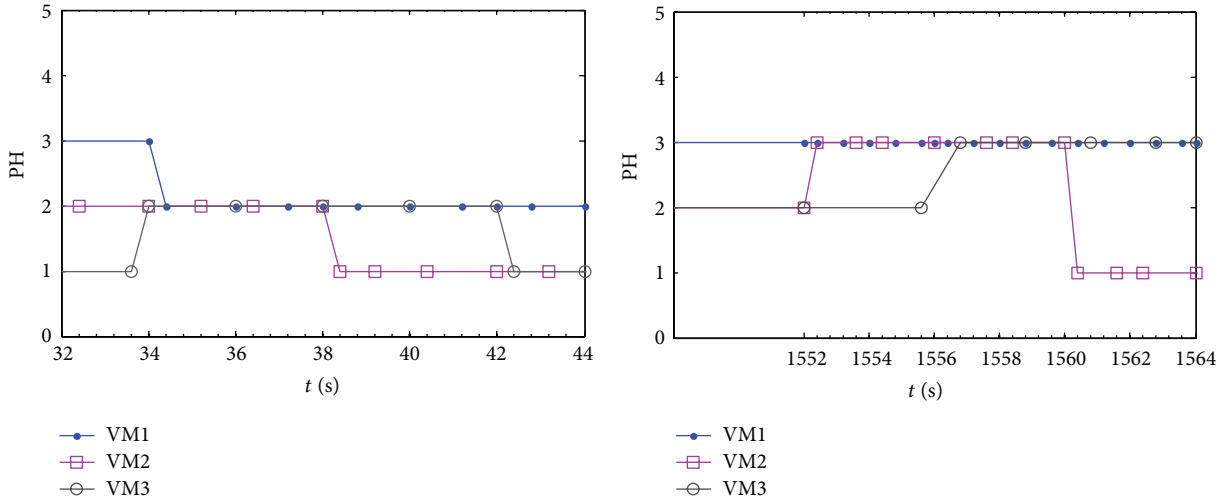
Then, we optimize the template parameter by using GA, PSO, BSPSO, and sort algorithm. In order to analyze the superiority of the proposed algorithm quantitatively, we define an evaluation criteria formula as (19) by using the number of iterations and fitness variance

$$\sigma^2 = \sum_{i=1}^{n} \left(\frac{f_i - f_{\text{avg}}}{f} \right), \tag{19}$$

where f is a normalization factor and can be taken as arbitrary values. f_i represents the fitness of the particle whose number is i, and f_{avg} denotes the average fitness of the particle swarm. The simulation result of the quantitative evaluation of the optimal solution efficiency is shown in Figure 4.

As depicted in Figure 4, the convergence speed of GA-based parameter optimization algorithm is faster than that of sort-based algorithm, and the performance of convergence speed of PSO algorithm is better than GA, but when the PSO algorithm finds a local optimal solution, it stops doing the searching, and sinks into the state of premature convergence. In the meantime, by adding the BS operating, the BSPSO model is slower than PSO in the first period of time, but it can jump out of the premature convergence and find the global optimum eventually.

5.2. The Experimentation Results of VMMD Based on CNN. This experimental environment contains four PHs, each host



(a) The result of VMMD from 32 s to 44 s

(b) The result of VMMD from 1552 s to 1564 s

Figure 6: The result of VMMD based on CNN in different times.

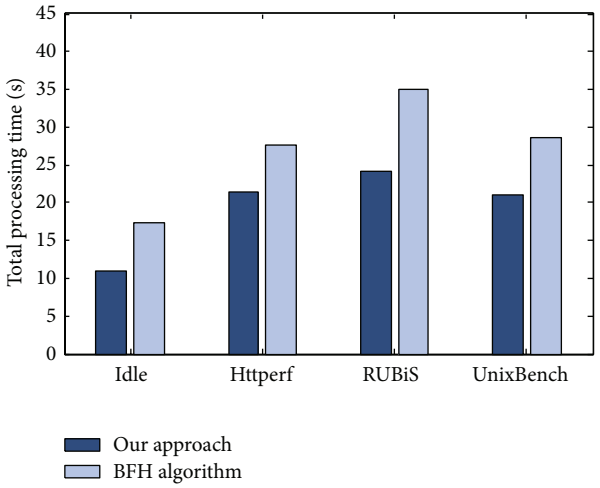


Figure 7: Total processing time of VMMD for different workloads.

consists of Intel (R) Core (TM) 2Duo E8400@ 3.0 GHz, 16 GB memory and is divided into a number of VMs with different configurations. The PH4 acts as VMM and connected with PH1, 2, and 3 with NFS server by 1 Gbps network bandwidth line. The virtualization software is Xen4.1.2 based on Linux platform.

By "top" order, we obtain the extent of access to a resource, such as memory allocated or CPU allocated to a VM. In 35 minutes, the CPU percentage use of PHs is shown in Figure 5 and when the sum of CPU percentage use of VMs on each PH exceeds 100%, like PH1 and PH2, the VMs instantiated on that PH will migrate to other PMs because of the violation of SLA.

By "public Map adjust VM Position (String umid)/∗@ param VM Name" order, we choose three VMs instantiated on different PH which are running EPA-HTTP benchmark tests, and the migration detection process based on CNN is shown in Figure 6.

As depicted in Figure 6(a), in the first period time, the state (memory pages) of VM3 is transferred from PH1 to PH2, and then the memory pages of VM1 migrate from PH3 to PH2; that is, VM1 and VM3 will suspend; they copy all its pages and then resume the VMs on the target machine PH2. The migration downtime is proportional to the size of the VMs and network resources available for state transfer. When the physical resources can not satisfy the need of VMs executed on PH2, each of which is self-contained with its own operating system, the targets of VM2 and VM3 will transfer from PH2 to PH1 in order and meet the sequence constraints. Similarly, Figure 6(b) also verifies the live VM migration policy satisfying the sequence constraints and SLA. The new PM can be added to offset the load of overloaded PM by migrating VM2 from PH3 to PH1. Likewise, hosting new VMs may result in future overloads of PH1, which will cause the migration requires to be triggered.

Figure 7 shows the total processing time of VMMD between our approach and traditional best fit heuristic algorithm (BFH) [22] for different workloads in 30 minutes. Compared with BFH algorithm, our approach reduced the processing time by 36.78%, 22.1%, 30.86%, and 26.48% for the workload of dynamic application (Idle, Httperf, RUBiS, and UnixBench), respectively. The reason is that the migration policy based on BFH should determine when a PH is considered being underloaded; hence, it becomes a good candidate for hosting VMs that are being migrated from overloaded PHs, that cause the migration downtime of dynamic application to be extremely long. Whereas our detection approach only executes iterations to perform the process of the suspending and copy phase, it is a feasible method with satisfying lightweight performance and global live monitoring advantage.

6. Conclusions

In this paper, we have proposed a live VMMD model based on CNN, which can monitor the security of target operating systems and it also provides the ability to inspect the VMs'

state. The migration procedure can be emerged as locally connected, nonlinear processor arrays, while the outputs reach their steady state values at an equilibrium point which represents a desirable feature in view of VLSI hardware implementations of real time networks. On the basis of analyzing the Local Rules 1 and 2 and the global rule, the parameter relationship can be mapped as a COP. Then, BSPSO algorithm has been designed to find out better optimum, avoiding the PSO algorithm trapping into local optimum. Experiments are carried out to demonstrate the performance of the proposed model, and the comparative results show that the proposed model exhibits superior performance with shorter processing time compared with BFH algorithm.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (no. 61121061), the National Natural Science Foundation of China (no. 61161140320), and the National Key Technology R&D Program (2012BAH37B05).

References

[1] K. J. Ye, H. Zh. Wu, X. H. Jiang, and Q. M. He, "Power management of virtualized cloud computing platform," *Chinese Journal of Computers*, vol. 6, pp. 1262–1285, 2012.

[2] S. Sangeeta and C. Meenu, "A technical review for efficient virtual machine migration," in *Proceedings of the International Conference on Cloud and Ubiquitous Computing and Emerging Technologies*, pp. 20–25, 2013.

[3] L. O. Chua and L. Yang, "Cellular neural networks: applications," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, pp. 1273–1290, 1988.

[4] I. Rodero, J. Jaramillo, A. Quiroz, M. Parashar, and F. Guim, "Towards energy-aware autonomic provisioning for virtualized environments," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC '10)*, pp. 320–323, June 2010.

[5] S. Parmaksizoğlu, E. Günay, and M. Alçi, "Determining clonning templates of CNN via MOGA: edge detection," in *Proceedings of the IEEE 19th Signal Processing and Communications Applications Conference (SIU '11)*, pp. 758–761, Antalya, Turkey, April 2011.

[6] L. Li, G. Ma, and X. Du, "New method of horizon recognition in seismic data," *IEEE Geoscience and Remote Sensing Letters*, vol. 9, no. 6, pp. 1066–1068, 2012.

[7] L. Wan, X. Liu, T.-T. Wong, and C.-S. Leung, "Evolving mazes from images," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 2, pp. 287–297, 2010.

[8] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, "Dynamic resource management using virtual machine migrations," *IEEE Communications Magazine*, vol. 50, no. 9, pp. 34–40, 2012.

[9] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *Journal of IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1366–1379, 2013.

[10] J. J. Liu, G. L. Chen, and X. Ch. Hu, "Virtual machine migration scheduling strategy based on load characteristic," *Computer Engineering*, vol. 37, no. 17, pp. 276–278, 2011.

[11] W. Liu and T. Fan, "Live migration of virtual machine based on recovering system and CPU scheduling," in *Proceedings of the 6th IEEE Joint International Information Technology and Artificial Intelligence Conference (ITAIC '11)*, pp. 303–307, Chongqing, China, August 2011.

[12] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "Improving virtual machine migration in federated cloud environments," in *Proceedings of the 2nd International Conference on Evolving Internet (Internet '10)*, pp. 61–67, Valencia, Spain, September 2010.

[13] Ch. Chen, H. X. Zhang, L. Zh. Yu, Y. Fan, and L. N. Liu, "A new live virtual machine migration strategy," in *Proceedings of the International Symposium on Information Technology in Medicine and Education (ITME '12)*, pp. 173–176, 2012.

[14] C.-H. Hsu, S.-C. Chen, C.-C. Lee et al., "Energy-aware task consolidation technique for cloud computing," in *Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom '11)*, pp. 115–121, Athens, Greece, December 2011.

[15] Y. Luo, B. Zhang, X. Wang, Z. Wang, Y. Sun, and H. Chen, "Live and incremental whole-system migration of virtual machines using block-bitmap," in *Proceedings of the IEEE International Conference on Cluster Computing (CCGRID '08)*, pp. 99–106, Tsukuba, Japan, October 2008.

[16] H. Zh. Li, W. Luo, X. J. Lu, and J. W. Yin, "A live migration strategy for virtual machine based on performance predicting," in *Proceedings of the International Conference on Computer Science and Service System*, pp. 72–76, 2012.

[17] L. Zhang, Y. B. Bai, and X. Wei, "A tracing approach to process migration for virtual machine based on multicore platform," in *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP '10)*, vol. 6081 of *Lecture Notes in Computer Science*, pp. 391–403, 2010.

[18] H. K. Liu, H. Jin, X. F. Liao, L. T. Hu, and Ch. Yu, "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 (HPDC '09)*, pp. 101–110, June 2009.

[19] R. H. Shang, L. Ch. Jiao, X. Ch. Hu, and J. J. Ma, "Modified immune clonal constrained multi-objective optimization algorithm," *Journal of Software*, vol. 23, no. 7, pp. 1773–1786, 2012.

[20] X. L. Li, G. Ch. Guo, X. Y. Li, and H. M. Hua, "A constraint optimization based mapping method for virtual network," in *Proceedings of the International Conference on Graphic and Image Processing (ICGIP '12)*, vol. 49, pp. 1601–1610, 2012.

[21] K. K. Wang, Q. Lv, H. Q. Zhao, and W. Zhang, "Hybrid algorithm for solving complex constrained optimization problems based on PSO and ABC," *Systems Engineering and Electronics*, vol. 34, no. 6, pp. 1193–1199, 2012.

[22] T. Mofolo and R. Suchithra, "Heuristic based resource allocation using virtual machine migration: a cloud computing perspective," *International Refereed Journal of Engineering and Science*, vol. 2, no. 5, pp. 40–45, 2013.