

Research Article

An Anomaly Detection Algorithm of Cloud Platform Based on Self-Organizing Maps

Jun Liu,¹ Shuyu Chen,² Zhen Zhou,¹ and Tianshu Wu¹

¹College of Computer Science, Chongqing University, Chongqing 400044, China

²College of Software Engineering, Chongqing University, Chongqing 400044, China

Correspondence should be addressed to Jun Liu; liujuncqcs@163.com

Received 24 November 2015; Accepted 6 March 2016

Academic Editor: Yassir T. Makkawi

Copyright © 2016 Jun Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Virtual machines (VM) on a Cloud platform can be influenced by a variety of factors which can lead to decreased performance and downtime, affecting the reliability of the Cloud platform. Traditional anomaly detection algorithms and strategies for Cloud platforms have some flaws in their accuracy of detection, detection speed, and adaptability. In this paper, a dynamic and adaptive anomaly detection algorithm based on Self-Organizing Maps (SOM) for virtual machines is proposed. A unified modeling method based on SOM to detect the machine performance within the detection region is presented, which avoids the cost of modeling a single virtual machine and enhances the detection speed and reliability of large-scale virtual machines in Cloud platform. The important parameters that affect the modeling speed are optimized in the SOM process to significantly improve the accuracy of the SOM modeling and therefore the anomaly detection accuracy of the virtual machine.

1. Introduction

As Cloud computing applications become increasingly mature, more and more industries and enterprises are deploying increasing numbers of applications within Cloud platforms, in order to improve efficiencies and on-demand services where resources are limited. Virtual machines for computing and resource storage are core to a Cloud platform and are essential to ensure normal operation of various businesses [1, 2]. However, as the number of applications increases, the scale of the Cloud platform is expanding. Resource competition, resource sharing, and load balancing within the Cloud platform reduce the stability of virtual machines, which leads directly to a decrease in the reliability of the entire Cloud platform [3–7]. Therefore, anomaly detection of virtual machines is an important method for durable and reliable operation on a Cloud platform.

At present, the main methods of virtual machine anomaly detection on Cloud platforms are to collect system operation logs and various performance metrics of the virtual machine status and then determine the anomaly using anomaly detection methods such as statistics, clustering, classification, and nearest neighbor.

The statistical anomaly detection method is a statistical method based on a probabilistic model. This method makes certain assumptions about the conditions [8]. However, in real Cloud platforms, the distribution of data is usually unpredictable, which means that the statistics-based method has low detection rates and thus may be unsuitable. Clustering-based methods group similar virtual machines states together and consider any states which are distant from the cluster center to be abnormal [9, 10]. Since this method does not need a priori knowledge of the data distribution, its accuracy is better than the statistics-based method. However, it is difficult to choose a reasonable clustering algorithm in clustering-based methods. Self-Organizing Maps (SOM) [11, 12], *k*-means [13, 14], and expectation maximization [15] are three commonly used clustering algorithms in anomaly detection. The classification-based algorithm mainly includes neural networks [16, 17], Bayesian networks [18, 19], and support vector machines [20–22]. The main drawback of these algorithms is the high training cost and the complexity of the implementation. The neighbor-based algorithm detects anomalies based on clustering or the similarity of the data. However, the main disadvantage of this algorithm is that the

recognition rate decreases when the normal dataset that is being detected does not have enough neighbors.

In a Cloud environment, the performance and running status of the virtual machine is represented mainly by the performance metrics. The performance metrics include five primary metrics: CPU, memory, disk, network, and process [23]. These metrics can determine whether a virtual machine is abnormal. Reference [23] has a more detailed explanation of the performance metrics of virtual machine.

This paper proposes an SOM-based anomaly detection algorithm which is based on determining the various performance metrics of each virtual machine. This algorithm is different from traditional strategies in that the detection domains of the virtual machines with similar running environments are divided and each domain is trained iteratively in the SOM network. This enables reasonable adaptation of the training of the large-scale virtual machines in the Cloud platform and overcomes the shortcomings of traditional methods where each virtual machine is treated as a training sample. In addition, two important parameters of the SOM network training are optimized, which greatly reduce the training time of the SOM network and the performance metrics of the virtual machine and enhances the efficiency and the accuracy of anomaly detection of the virtual machine in the Cloud platform.

Various experiments were conducted in order to verify the efficiency and accuracy of the SOM-based anomaly detection algorithm. The results show that the sample training speed and detection accuracy are significantly improved by the proposed algorithm.

The rest of this paper is organized as follows. Section 2 describes existing anomaly detection methods. Section 3 describes the SOM-based virtual machine anomaly detection algorithm. Section 4 shows the performance evaluation. And, finally, Section 5 lists the conclusions derived from the experiment.

2. Related Work

Current anomaly detection methods are mainly based on classification, clustering, statistics, and nearest neighbor methods [24]. These methods will now be introduced.

The classification-based method obtains a classifier model from a set of selected data and then uses the model to classify new data [25, 26]. Shin and Kim proposed a hybrid classification method that combines the One-Class SVM [27, 28] hybrid classification method with the nearest mean classifier (NMC) [29]. The highly flexible nonlinear correlation model can be easily classified by the nonlinear kernel function in this method [30–32]. This method introduces a feature subset selection algorithm, which not only reduces the number of classification dimensions, but also improves the performance of the classifier. However, the main disadvantage of this method is slow training and potential for misclassification.

The clustering-based method is an unsupervised learning method [26, 33]. SNN [34], ROCK [35], and DBSCAN [36] are three typical clustering-based anomaly detection methods. All of these three methods assume that normal samples

are within a single cluster within the dataset, and abnormal samples are outside any cluster. However, if a cluster is formed by the anomaly data after a period of clustering, then the anomalies cannot be recognized properly. Additionally, it is important for the clustering that the width of the cluster is accurately selected. The advantages of the clustering-based approach are that a priori knowledge of the data distribution is not required and it can be used for incremental modeling. For example, for anomaly detection of virtual machines, a newly collected virtual machine sample can be analyzed by a model already known for anomaly detection.

A typical nearest neighbor based approach is proposed by Breunig et al. [37] using a local outlier factor for the data abnormality detection. Any data that requires analysis is associated with a local outlier factor, which is the ratio of the average local density of the k nearest neighbors to the data itself. The local density is the volume of data-centric spheres of the k smallest neighbors divided by k . If data is abnormal, then its local density should be significantly different than the local density of its nearest neighbors.

The statistics-based approach is an earlier anomaly detection method, which is usually based on an assumption that an anomaly is an observation point that is not generated by an assumed model and is partly or completely irrelevant [37]. Ye and Chen [24, 38] used χ^2 statistics to detect anomalies in the operating system. Assuming that normal data under training is subject to a multivariate normal distribution, then the χ^2 is

$$\chi^2 = \sum_{i=1}^n \frac{(X_i - E_i)^2}{E_i}, \quad (1)$$

where X_i is the observed value of the i th variable, E_i is the expected value of the i th variable (obtained from the training data), and n is the number of variables. A large value of χ^2 represents an anomaly in the observed samples.

3. SOM-Based Virtual Machine Anomaly Detection Algorithm

In a Cloud platform, virtual machines with a similar running environment have similar system performances. A SOM-based virtual machine anomaly detection algorithm is aimed at Cloud platforms that have a large number of virtual machines. In this paper, we partition virtual machines with similar running environments; that is, we assign a set of virtual machines with similar properties to the same detection field. This avoids the requirement for SOM network modeling for every single virtual machine, significantly reducing the modeling time and the training cost. For instance, when the proposed method is not used, 100 SOM network models need to be built for 100 virtual machines; however with the proposed method, 100 virtual machines with similar running environments only need one SOM network model to be built. In addition, a SOM network can be trained more accurately by collecting 100 samples than by training using one sample only.

After partition of the virtual machines, SOM network training is used in every domain. In this paper, the two most

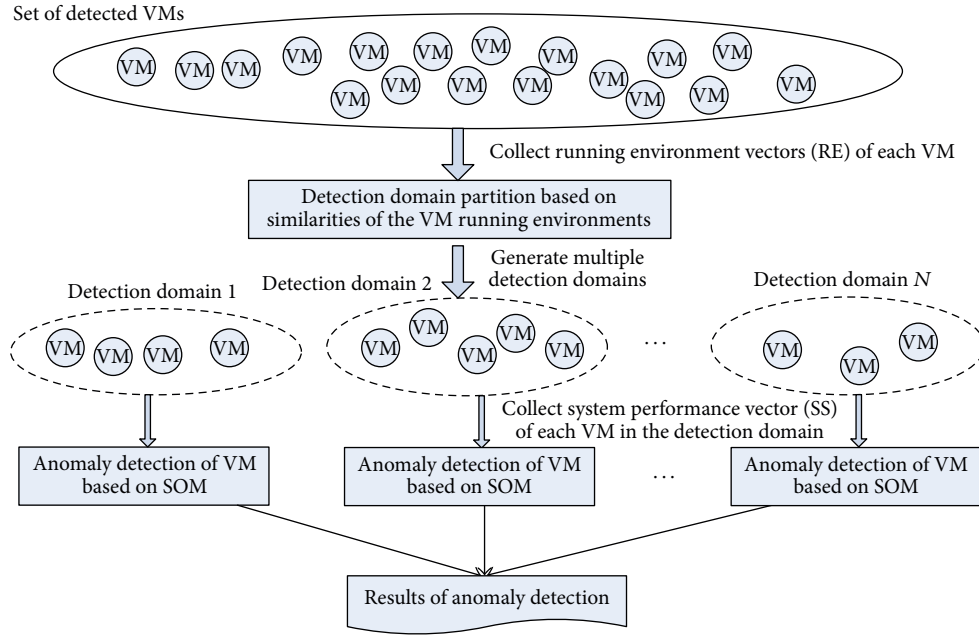


FIGURE 1: SOM anomaly detection logic diagram.

important parameters of training, width and learning-rate factor, are optimized to enhance the training speed. The flow chart of the anomaly detection algorithm is shown in Figure 1.

3.1. SOM-Based State Modeling of the Virtual Machine. Because prior knowledge of similar performance for virtual machine classification is unknown, the k -medoids method is used in this paper for initial classification; that is, the VMs on the Cloud platform are divided into multiple detection domains. The reason the k -medoids method is chosen is that, compared with the k -means algorithm, k -medoids is less susceptible to noise.

The SOM network is generated in each detection domain using the SOM algorithm. The network is constructed as a two-dimensional ($N \times N$) neuron array. Each neuron can be represented as n_{ij} , $i = 1, 2, 3, \dots, N$, and each neuron is related to a weight vector, which is defined as $W_{ij}(w_1, w_2, w_3, \dots, w_m)$. j is the column subscript. The dimensions of a weight vector m are the same as the dimensions of the training set for training its SOM network. The training set used in this paper includes the CPU utilization performance which reflects the running state of the virtual machine, its memory utilization, and its network throughput. These performance metrics are described by a vector defined as $SS(ss_1, ss_2, ss_3, \dots, ss_m)$.

The modeling of a specific virtual machine-based detection domain in SOM requires periodic measurements and adequate collection of the training data (performance x). The collected performance vector $SS \in R_m$ can be considered to be a random variable within the performance sample space. The VM performance samples collected within a certain time series can be expressed as SS_t (where $t = 1, 2, 3, \dots, n$). The iterative training of the samples collected within this time series is the modeling process of the SOM virtual machine

Therefore, the detection domain modeling algorithm can be summarized as follows.

Step 1 (initialization of the SOM network). SOM neurons are represented by a weight vector ($W_{ij}(0)$, $i, j = 1, 2, 3, \dots, N$), where i and j indicate the location of the neurons in the SOM network. In this paper, the weight vector ($W_{ij}(0)$) is initialized randomly in the SOM network.

Step 2 (defining the training space of the SOM network for training sample SS_t). When a training sample SS_t at time t is added to the SOM network, the most suitable neuron needs to be found to be the training center of the neighborhood. For SS_t at time t , the most suitable neuron C can be found using (2), and C will be the training center in the SOM network after SS_t is added:

$$C = \begin{cases} \arg \min_{(i,j)} \{ \|SS_t - W_{ij}(0)\| \}, & t = 1 \\ \arg \min_{(i,j)} \{ \|SS_t - W_{ij}(t-1)\| \}, & t = 2, 3, \dots \end{cases} \quad (2)$$

After the training center C is defined using (2), we need to set the training neighborhood. According to the definition of SOM, to ensure convergence of the training process of the SOM network, the training neighborhood can be defined as

$$P = H_C^{(i,j)}(t, \|l_C - (i, j)\|), \quad (3)$$

where P is a function of the training neighborhood that is a monotonically decreasing function of parameter $\|l_C - (i, j)\|$ and training iterations t ; l_C is the coordinate vector of the training center C in the SOM network; and (i, j) is the coordinate vector of neuron node n_{ij} in the SOM network. Due to its effective smoothing, a Gaussian function is used

as the neighborhood training function in this paper, which is defined as follows:

$$H_C^{(i,j)} = \alpha(t) \cdot \exp\left(-\frac{\|l_C - (i,j)\|^2}{2\sigma^2(t)}\right). \quad (4)$$

In (4), $\alpha(t)$ represents the learning-rate factor, which determines the fitting ability of the SOM network for the training sample SS_t in the training process. $\sigma(t)$ represents the width of the neighborhood that determines the range of influence of a single training sample SS_t on the SOM network. According to SOM related theory, to ensure convergence of the training process, $\alpha(t)$ and $\sigma(t)$ should be both monotonically decreasing functions of the number of training iterations t .

Step 3 (SOM network training based on training sample SS_t). The training neighborhood has been defined in Step 2. The neurons, which are within the training domain of the SOM network, are trained based on the training sample SS_t according to (5). The fitting equation is defined as follows:

$$W_{ij}(t) = W_{ij}(t-1) + H_C^{(i,j)} \cdot [SS_t - W_{ij}(t-1)]. \quad (5)$$

After the training process is completed using (5), the convergence of the training process needs to be verified. The process is convergent if every neuron associated with its weight vector in a SOM network is stabilized. The method is described in detail below.

Assume that there is a neuron n_{ij} in the SOM network and the time index of its latest training sample is $t_l^{(i,j)}$. Meanwhile, assume that there is a sufficiently small real number ε and that convergence of the training process of the SOM network can be checked using the following:

$$d(t) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \left\| W_{ij}(t_l^{(i,j)}) - W_{ij}(t_l^{(i,j)} - 1) \right\| \leq \varepsilon. \quad (6)$$

In (6), $d(t)$ represents the average deviation between the latest fitting state and the previous value for every neuron with a weight vector in the SOM network after t training samples are used in a training process. Obviously, when $d(t) < \varepsilon$, the neurons W_{ij} with a weight vector are stabilized, indicating that the iterative training process can be stopped. When $d(t) > \varepsilon$, further collection of the training samples is required, and Steps 2 and 3 need to be repeated.

3.2. Parameter Setting in the SOM-Based Modeling Process.

The SOM network modeling process is an iterative fitting process that mainly consists of two stages: the initial ordered stage and the convergence stage. There are two important parameters in the training neighborhood function $H_C^{(i,j)}$: the width of the training neighborhood $\sigma(t)$ and the learning-rate factor $\alpha(t)$. Correct setting of these two parameters plays an important role in preventing the SOM network training from getting trapped in a metastable state. The processes for setting these two parameters are as follows.

(1) *Setting the Width of the Training Neighborhood $\sigma(t)$* . Based on the principle of SOM, $\sigma(t)$ is a monotonically decreasing

function of t . At the beginning of the training process, the value of $\sigma(t)$ should be set properly so that the radius of the neighborhood defined by $H_C^{(i,j)}$ can reach at least half the diameter of the SOM network [39]. In this paper, the value is set to $N/2$.

Since $H_C^{(i,j)}$ is a monotonically decreasing function of $\|l_C - (i,j)\|$, it can be seen from (4) that, when other variables remain unchanged, the $H_C^{(i,j)}$ value is small if the neuronal node is distant from the training center. Additionally, if $H_C^{(i,j)}$ is smaller, it has a lower influence on the neuronal node n_{ij} in the fitting process. When the value of $H_C^{(i,j)}$ is small enough, the neuron node n_{ij} is unaffected. Therefore, although there is no clear boundary for the training neighborhood defined by the Gaussian function in this paper, the influential range of a single training sample SS_t on the training of the SOM network can still be limited.

Assume that f is a sufficiently small threshold of $H_C^{(i,j)}$. When $H_C^{(i,j)} < f$, the current iteration step has no influence on neuronal node n_{ij} , while when $H_C^{(i,j)} > f$, the current iteration step will influence n_{ij} .

Therefore, when $t = 1$ at the beginning of the SOM network training process, the lower bound of $\sigma(t)$ can be determined based on the threshold f and (4). The detailed derivation process is shown as follows.

When $t = 1$, assume that $\|l_C - (i,j)\| = N/2$, and the lower bound of $\sigma(1)$ is then determined by the following inequality derivation process:

$$\begin{aligned} \alpha(1) \cdot \exp\left(-\frac{(N/2)^2}{2\sigma^2(1)}\right) &\geq f \implies \\ \ln \alpha(1) - \frac{(N/2)^2}{2\sigma^2(1)} &\geq \ln f \implies \\ \frac{(N/2)^2}{2\sigma^2(1)} &\leq \ln \frac{f}{\alpha(1)} \implies \\ \sigma^2(1) &\geq \frac{N^2}{8 \cdot \ln(f/\alpha(1))} \implies \\ \sigma(1) &\leq -\frac{N}{2\sqrt{\ln(f/\alpha(1))}}, \\ \sigma(1) &\geq \frac{N}{2\sqrt{\ln(f/\alpha(1))}}, \\ \because \sigma(1) &> 0, \\ \therefore \implies \sigma(1) &\geq \frac{N}{2\sqrt{\ln(f/\alpha(1))}}. \end{aligned} \quad (7)$$

Based on this derivation, the lower bound of $\sigma(1)$ can be determined by (7), where the threshold $f = 0.05$ in this paper. The following discussion will describe the value of $\alpha(1)$ used

for setting $\alpha(t)$. According to (7), $\sigma(t)$ in $H_C^{(i,j)}$ of the initial ordered stage can be defined as follows:

$$\sigma(t) = \frac{N}{2\sqrt{\ln(f/\alpha(1))^2}} \cdot \exp\left(-\frac{t-1}{t}\right), \quad (8)$$

$$t = 1, 2, 3, \dots, 1000.$$

When the iteration of the SOM network training is gradually converging, the size of the training neighborhood defined by $H_C^{(i,j)}$ should be constant and can cover the nearest neighborhood of the training center C in the SOM network. In this paper, the nearest neighborhood, that is, the nearest four neurons around neuron C in all four directions (up, down, left, and right) in the SOM network, is shown in Figure 2.

(2) *Setting the Learning-Rate Factor $\alpha(t)$.* Since $\alpha(t)$ is a monotonically decreasing function of t , the range of $\alpha(t)$ is $0.2 < \alpha(t) < 1$ in the initial ordered stage of the SOM training process, and $0 < \alpha(t) < 0.2$ in the convergent stage of the SOM training process. Then $\alpha(t)$ can be set to

$$\alpha(t) = \begin{cases} \exp\left(-\frac{t-1}{t}\right), & t = 1, 2, 3, \dots, 1000 \\ 0.2 \cdot \exp\left(-\frac{t-1}{t}\right), & t > 1000. \end{cases} \quad (9)$$

3.3. VM Anomaly Recognition Based on SOM Model. The modeling method of VM status based on SOM is described sufficiently in the previous section. In this section, we will describe the recognition of an anomaly using the trained SOM network. After several rounds of fitting iterations, the SOM network can be used to effectively discover the normal state of virtual machines. The normal state is represented by neurons with weight vectors in the SOM network. In other words, a neuron associated with weight vectors in the SOM network can be used to describe whether a class of similar virtual machines is normal.

In order to check whether the current state of a VM is an anomaly on a Cloud platform, we can compare the current running performance of virtual machines with the neurons with weight vectors in the SOM network. In this paper, Euclidean distance is used to determine similarity. If the current state is similar to one of the neurons with weight vectors (assuming that the probability of anomaly is less than a given threshold t), the virtual machine will be identified to be normal; otherwise it will be considered to be abnormal.

Let VM_x represent a virtual machine on a Cloud platform. The corresponding SOM network of VM_x is defined as $SOM(VM_x)$. The weight vector of each neuron can be represented as W_{ij}^S , after the training iterations have finished. The currently measured performance value of VM_x is SS , and the abnormal state of VM_x is $VmStatus(VM_x)$. Then the

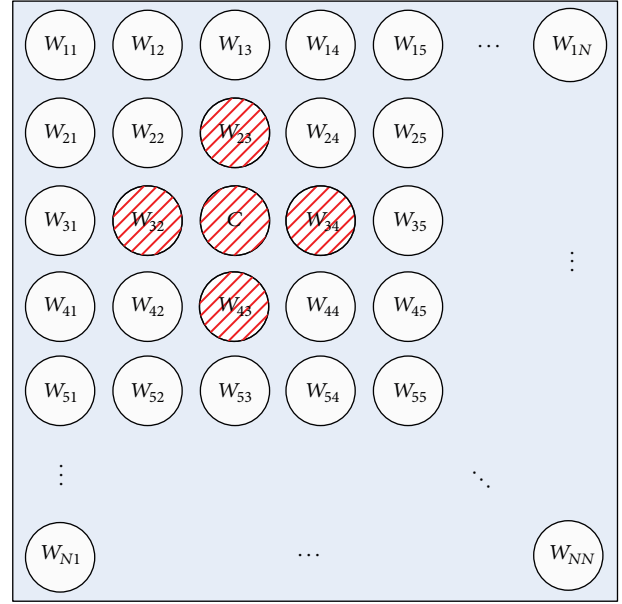


FIGURE 2: Nearest neighborhood of neuron node C .

abnormal state of the virtual machine can be determined by the following equation:

$$\text{anomaly}(VM_x) = \begin{cases} \text{true}, & \min \{\|SS - W_{ij}^S\| \mid i, j = 1, 2, 3, \dots, N\} \geq \delta \\ \text{false}, & \min \{\|SS - W_{ij}^S\| \mid i, j = 1, 2, 3, \dots, N\} < \delta \end{cases} \quad (10)$$

in which δ is a sufficiently small constant.

4. Experimental Results and Analysis

4.1. Experimental Environment and Setup. In this paper, the experimental Cloud platform is built on an open source Cloud platform *OpenStack* [40, 41]. The operating system CentOS 6.3 is installed on the physical servers for the running virtual machines, on which the hypervisor Xen-3.2 [42, 43] is installed. The operating system CentOS 6.3 is also installed on the physical servers for running the Cloud management program, on which the Cloud management components *OpenStack* are installed. 100 virtual machines were deployed on this Cloud platform.

The performance metrics of the virtual machines in this experiment were collected by tools such as *libxenstat* and *libvirt* [44, 45]. For the fault injection method, we used tools to simulate system failures: memory leak, CPU Hog, and network Hog [46–48].

4.2. Experimental Program and Results

4.2.1. First Set of Experiments. The impact of the SOM network, training neighborhood width, and learning-rate factor values on the performance of the anomaly detection

TABLE 1: The impact of SOM net size on the detection accuracy.

Size of SOM net	Accuracy rate (%)
8 × 8	≈96.3
13 × 13	≈97.9
18 × 18	≈97.5
20 × 20	≈97.7
24 × 24	≈97.8

TABLE 2: The impact of the initial training neighborhood size on the accuracy of SOM.

The initial width for the training neighborhood	Detection accuracy
0.5 dsn	≈97.8
0.4 dsn	≈93.1
0.3 dsn	≈90.4
0.2 dsn	≈89.1
0.1 dsn	≈73.7

Dsn indicates the diameter of the SOM network.

TABLE 3: The impact of the initial value of the learning-rate factor on the accuracy of SOM.

Initial learning-rate factor	Detection accuracy
1	≈98
0.9	≈95.6
0.8	≈93.3
0.7	≈90.7
0.6	≈88.6

mechanism of the SOM-based dynamic adaptive virtual machine was evaluated.

Training Stage. Firstly, several virtual machines were selected from 100 virtual machines. One fault was then randomly selected (a memory leak, CPU Hog, or network Hog) and then injected. 1000 virtual machine system performance measurements were collected as training samples for the model training during 10 rounds (one second per round), on the 100 virtual machines.

Anomaly Detection Stage. In order to simulate an anomaly in the objects under detection, one of the three faults was randomly injected in the 100 virtual machines. The anomalies in each of the 100 virtual machines were then detected based on the trained model. The detection results were recorded.

Several sets of experimental results with different parameter values were obtained. It should be noted that the same fault was injected in each experiment to exclude unnecessary variables.

The experimental results are shown in Tables 1, 2, and 3.

As can be seen from Table 1, there is no obvious change in accuracy using the proposed detection method for different SOM network sizes, which means that the proposed anomaly detection method is not affected by the size of the SOM network.

TABLE 4: The impact of SOM net size on the detection accuracy.

Size of SOM net	Accuracy rate (%)
8 × 8	≈95.8
13 × 13	≈97.1
18 × 18	≈96.7
20 × 20	≈96.9
24 × 24	≈97.1

TABLE 5: The impact of the initial training neighborhood size on the accuracy of SOM.

The initial width for the training neighborhood	Detection accuracy
0.5 dsn	≈97.1
0.4 dsn	≈92.5
0.3 dsn	≈90.1
0.2 dsn	≈89.4
0.1 dsn	≈73.1

TABLE 6: The impact of the initial value of the learning-rate factor on the accuracy of SOM.

Initial learning-rate factor	Detection accuracy
1	≈97.5
0.9	≈95.4
0.8	≈93.1
0.7	≈89.9
0.6	≈88.1

As can be seen from Table 2, the size of the initial trained neighborhood has a significant impact on the detection accuracy. The main reason is that if the training size is too small, it may cause a metastable state in the training process, and further training iterations are required to achieve real steady state.

As can be seen from Table 3, as the initial value of the learning-rate factor decreases, the accuracy of the abnormality detection significantly decreases. The reason is that if the initial value of the learning-rate factor is too small, the contribution of each training sample in the SOM network training is small too. Thus the fitting ability of the SOM network to detect an object is not sufficient, which leads to poor quality of model training, hence decreasing the accuracy of the SOM network-based anomaly detection.

Analysis of the first set of experiments shows that better anomaly detection results can be obtained in DA_SOM when the parameters are set as follows: SOM network size = 13 × 13, initial size of training neighborhood = 0.5 dsn, and initial value of learning-rate factor = 1.

The above experiments have been carried out on the training data set. To further demonstrate the effectiveness of the proposed algorithm, the algorithm is tested on the untrained anomaly set (disk Hog).

The experimental results about disk Hog are shown in Tables 4, 5, and 6.

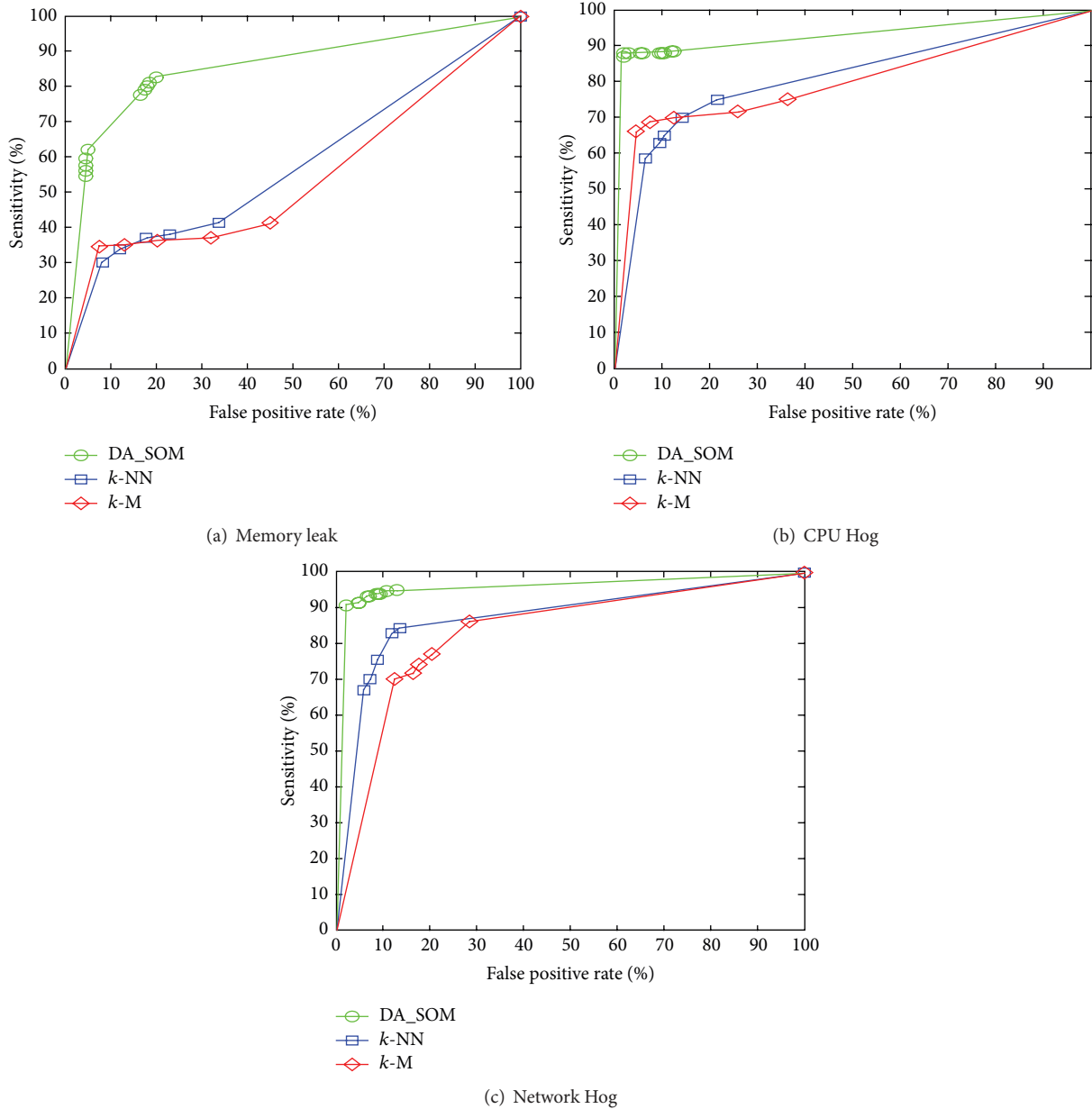


FIGURE 3: Comparison of the three anomaly detection algorithms: DA_SOM, k -NN, and k -M.

As can be seen from Tables 4, 5, and 6, the accuracy of the proposed algorithm still has better accuracy in the untrained data set. The impact of three parameters (som net size, training neighborhood width, and learning-rate factor) on the accuracy is similar with the previous experiments.

4.2.2. *Second Set of Experiments.* The objective of this set of experiments was to evaluate the effect of the VM anomaly detection mechanism based on SOM (represented by DA_SOM in the following sections). In order to compare this with other approaches, we use two typical unsupervised anomaly detection techniques in the experiments: (1) k -nearest neighbor based anomaly detection technique (called k -NN) where prior training of the anomaly identification

model is not required; (2) cluster-based anomaly detection technique (called k -M) where training of the anomaly identification model is required in advance.

Several experiments for different techniques and different parameters with the same aforementioned configuration and experimental procedure are applied to obtain the corresponding results. It should be noted that, since the training process is not required for the k -NN technique, it started directly in the abnormality detection stage. In addition, to ensure comparability, the training process of the clustering-based method is the same as the proposed method, where an anomaly detection model is built for 100 virtual machines and the training data set is the same as training SOM. Experimental results are shown in Figure 3.

Figure 3 shows that compared to the other two injected failures, the sensitivities of the three techniques to memory leak failure are relatively low. The main reason is that an anomaly does not immediately appear on the failed object when there is fault introduced by a memory leak. It takes some time for this fault to accumulate to eventually cause an obvious abnormality. The consequence of this is that detection systems tend to mistake these objects with an anomaly as normal. In contrast, faults caused by CPU Hog and network Hog events will immediately lead to an abnormal state within the fault object thus minimizing misjudgments, which enhances the sensitivity of all three anomaly detection techniques, as shown in Figures 3(b) and 3(c).

Meanwhile, as shown in each subgraph of Figure 3, compared with the other two anomaly detection techniques, DA_SOM maintains a better balance between sensitivity and false alarm rate. In other words, with the same false alarm rate, the sensitivity of DA_SOM is better than that of the other two approaches, showing a strong performance in improving warning effect and reducing the false alarm rate.

Moreover, the computational complexity of DA_SOM is much lower than that of the k -NN in anomaly detection stage while the computational complexity of DA_SOM is equivalent to the k -M technique. Their complexity is constant with the detected object size and with the parameter k in the k -M technique. Meanwhile, during the model training stage, the training cost of k -M is higher than that of DA_SOM, for the same size of training data. The main reason is that iteration is required in k -M on the entire training data set (i.e., the cluster centers need to be updated, and the training data set needs to be reclassified according to the updated center point), while there is only one classification operation for each training sample in DA_SOM.

5. Conclusion

An anomaly detection algorithm based on SOM for the Cloud platform with large-scale virtual machines is proposed. The virtual machines are partitioned initially according to their similarity, and, then based on the results of initial partition, the SOM is modeled. The proposed method has a high training speed, which is not possible in traditional methods when there are a large number of virtual machines. We also optimized the two main parameters in the SOM network modeling process, which highly improved this process. The proposed method is verified on an incremental SOM anomaly detection model. The results showed strong improvements in detection accuracy and speed using the proposed anomaly detection method.

Competing Interests

The authors declare that they have no competing interests.

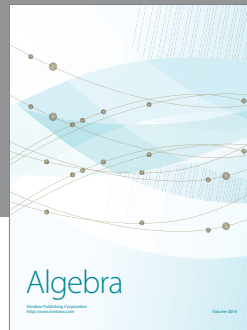

Acknowledgments

The work of this paper is supported by National Natural Science Foundation of China (Grants no. 61272399 and no. 61572090) and Research Fund for the Doctoral Program of Higher Education of China (Grant no. 20110191110038).

References

- [1] J. Li, Y. Cui, and Y. Ma, "Modeling message queueing services with reliability guarantee in cloud computing environment using colored petri nets," *Mathematical Problems in Engineering*, vol. 2015, Article ID 383846, 20 pages, 2015.
- [2] M. A. Rodriguez-Garcia, R. Valencia-Garcia, F. Garcia-Sanchez, and J. J. Samper-Zapater, "Ontology-based annotation and retrieval of services in the cloud," *Knowledge-Based Systems*, vol. 56, pp. 15–25, 2014.
- [3] C.-C. Chang, C.-Y. Sun, and T.-F. Cheng, "A dependable storage service system in cloud environment," *Security and Communication Networks*, vol. 8, no. 4, pp. 574–588, 2015.
- [4] W. He and L. Xu, "A state-of-the-art survey of cloud manufacturing," *International Journal of Computer Integrated Manufacturing*, vol. 28, no. 3, pp. 239–250, 2015.
- [5] A. F. Barsoum and M. Anwar Hasan, "Provable multicopy dynamic data possession in cloud computing systems," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 485–497, 2015.
- [6] J. Subirats and J. Guitart, "Assessing and forecasting energy efficiency on Cloud computing platforms," *Future Generation Computer Systems*, vol. 45, pp. 70–94, 2015.
- [7] S. Ding, S. Yang, Y. Zhang, C. Liang, and C. Xia, "Combining QoS prediction and customer satisfaction estimation to solve cloud service trustworthiness evaluation problems," *Knowledge-Based Systems*, vol. 56, pp. 216–225, 2014.
- [8] I. C. Paschalidis and Y. Chen, "Statistical anomaly detection with sensor networks," *ACM Transactions on Sensor Networks*, vol. 7, no. 2, article 17, 2010.
- [9] M. GhasemiGol and A. Ghaemi-Bafghi, "E-correlator: an entropy-based alert correlation system," *Security and Communication Networks*, vol. 8, no. 5, pp. 822–836, 2015.
- [10] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, NJ, USA, 1988.
- [11] M. Kourki and M. A. Riahi, "Seismic facies analysis from pre-stack data using self-organizing maps," *Journal of Geophysics and Engineering*, vol. 11, no. 6, Article ID 065005, 2014.
- [12] L. Feng and S. LiQuan, "Enhanced dynamic self-organizing maps for data cluster," *Information Technology Journal*, vol. 26, no. 1, pp. 70–81, 2009.
- [13] Z. Zhou, S. Chen, M. Lin, G. Wang, and Q. Yang, "Minimizing average startup latency of VMs by clustering-based template caching scheme in an IaaS system," *International Journal of u-and e- Service, Science and Technology*, vol. 6, no. 6, pp. 145–158, 2013.
- [14] L. Jing, M. K. Ng, and J. Z. Huang, "An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1026–1041, 2007.
- [15] R. Smith, A. Bivens, M. Embrechts, C. Palagiri, and B. Szymanski, "Clustering approaches for anomaly based intrusion detection," in *Proceedings of Intelligent Engineering Systems through Artificial Neural Networks*, pp. 579–584, 2002.
- [16] Y. Sani, A. Mohamedou, K. Ali, A. Farjamfar, M. Azman, and S. Shamsuddin, "An overview of neural networks use in anomaly intrusion detection systems," in *Proceedings of the IEEE Student Conference on Research and Development (SCORED '09)*, pp. 89–92, Serdang, Malaysia, November 2009.
- [17] G. P. Zhang, "Neural networks for classification: a survey," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 30, no. 4, pp. 451–462, 2000.

- [18] W. Tylman, "Anomaly-based intrusion detection using Bayesian networks," in *Proceedings of the International Conference on Dependability of Computer Systems*, pp. 211–218, Szklarska Poręba, Poland, 2008.
- [19] W. Pedrycz, V. Loia, and S. Senatore, "P-FCM: a proximity-based fuzzy clustering," *Fuzzy Sets and Systems*, vol. 148, no. 1, pp. 21–41, 2004.
- [20] G. Rätsch, S. Mika, B. Schölkopf, and K.-R. Müller, "Constructing boosting algorithms from SVMs: an application to one-class classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1184–1199, 2002.
- [21] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004.
- [22] B. Schölkopf, R. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Piatt, "Support vector method for novelty detection," in *Proceedings of the 13th Annual Neural Information Processing Systems Conference (NIPS '99)*, pp. 582–588, December 1999.
- [23] G. Wang, S. Chen, Z. Zhou, and M. Lin, "A dependable monitoring mechanism combining static and dynamic anomaly detection for network systems," *International Journal of Future Generation Communication and Networking*, vol. 7, no. 1, pp. 1–18, 2014.
- [24] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: a survey," *ACM Computing Surveys*, vol. 41, no. 3, article 15, 2009.
- [25] P. N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, Addison-Wesley, Reading, Mass, USA, 2005.
- [26] R. O. Duda, E. P. Hart, and D. G. Stork, *Pattern Classification*, Wiley-Interscience, New York, NY, USA, 2nd edition, 2000.
- [27] D. Shin and S. Kim, "Nearest mean classification via one-class SVM," in *Proceedings of the International Joint Conference on Computational Sciences and Optimization (CSO '09)*, pp. 593–596, Sanya, China, April 2009.
- [28] T.-S. Li and C.-L. Huang, "Defect spatial pattern recognition using a hybrid SOM-SVM approach in semiconductor manufacturing," *Expert Systems with Applications*, vol. 36, no. 1, pp. 374–385, 2009.
- [29] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.
- [30] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.
- [31] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, Cambridge, Mass, USA, 2000.
- [32] B. Liu, Y. Xiao, Y. Zhang, and Z. Hao, "An efficient approach to boost support vector data description," in *Proceedings of the 2012 International Conference on Cybernetics and Informatics*, vol. 163 of *Lecture Notes in Electrical Engineering*, pp. 2231–2238, Springer, New York, NY, USA, 2014.
- [33] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, New York, NY, USA, 1988.
- [34] L. Ertöz, M. Steinbach, and V. Kumar, "Finding topics in collections of documents: a shared nearest neighbor approach," in *Clustering and Information Retrieval*, vol. 11, pp. 83–103, 2003.
- [35] S. Guha, R. Rastogi, and K. Shim, "Rock: a robust clustering algorithm for categorical attributes," *Information Systems*, vol. 25, no. 5, pp. 345–366, 2000.
- [36] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, E. Simoudis, J. Han, and U. Fayyad, Eds., pp. 226–231, AAAI Press, Portland, Ore, USA, August 1996.
- [37] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, vol. 29, no. 2, pp. 93–104, 2000.
- [38] N. Ye and Q. Chen, "An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems," *Quality and Reliability Engineering International*, vol. 17, no. 2, pp. 105–112, 2001.
- [39] T. Kohonen, *Self-Organizing Maps*, Springer, New York, NY, USA, 1997.
- [40] J. M. Alcaraz Calero and J. G. Aguado, "MonPaaS: an adaptive monitoring platform as a service for cloud computing infrastructures and services," *IEEE Transactions on Services Computing*, vol. 8, no. 1, pp. 65–78, 2015.
- [41] D. Milošević, I. M. Llorente, and R. S. Montero, "OpenNebula: a cloud management tool," *IEEE Internet Computing*, vol. 15, no. 2, pp. 11–14, 2011.
- [42] H. Jin, H. Qin, S. Wu, and X. Guo, "CCAP: a cache contention-aware virtual machine placement approach for HPC cloud," *International Journal of Parallel Programming*, vol. 43, no. 3, pp. 403–420, 2015.
- [43] B. Egger, E. Gustafsson, C. Jo, and J. Son, "Efficiently restoring virtual machines," *International Journal of Parallel Programming*, vol. 43, no. 3, pp. 421–439, 2015.
- [44] Y. Cho, J. Choi, J. Choi, and M. Lee, "Towards an integrated management system based on abstraction of heterogeneous virtual resources," *Cluster Computing*, vol. 17, no. 4, pp. 1215–1223, 2014.
- [45] J. Li, Y. Jia, L. Liu, and T. Wo, "CyberLiveApp: a secure sharing and migration approach for live virtual desktop applications in a cloud environment," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 330–340, 2013.
- [46] Z. Xu, J. Zhang, and Z. Xu, "Melton: a practical and precise memory leak detection tool for C programs," *Frontiers of Computer Science*, vol. 9, no. 1, pp. 34–54, 2015.
- [47] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: an evaluation of the state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743–761, 2012.
- [48] Y.-J. Chiu and T. Berger, "A software-only videocodec using pixelwise conditional differential replenishment and perceptual enhancements," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 3, pp. 438–450, 1999.

Hindawi

Submit your manuscripts at
<http://www.hindawi.com>

