Cluster Computing
https://doi.org/10.1007/s10586-021-03385-2

A two-stage virtual machine abnormal behavior-based anomaly detection mechanism

Hancui Zhang[1] · Weida Zhou[1]

Received: 25 September 2020 / Revised: 5 June 2021 / Accepted: 3 August 2021

Abstract
Virtual machine abnormal behavior detection is an effective way to help cloud platform administrators monitor the running status of cloud platform to improve the reliability of cloud platform, which has become one of the research hotspots in the field of cloud computing. Aiming at the problems of high computational complexity and high false alarm rate in the existing virtual machine anomaly monitoring mechanism of cloud platform, this paper proposed a two-stage virtual machine abnormal behavior-based detection mechanism. Firstly, a workload-based incremental clustering algorithm is used to monitor and analyze both the virtual machine workload information and performance index information. Then, an online anomaly detection mechanism based on the incremental local outlier factor algorithm is designed to enhance detection efficiency. By applying this two-phase detection mechanism, it can significantly reduce the computational complexity and meet the needs of real-time performance. The experimental results are verified on the mainstream Openstack cloud platform.

Keywords Abnormal behavior detection · Reliability · Workload-based clustering · Local outlier factor

1 Introduction

Advanced in network and virtualization technology, cloud computing technologies and the exponential growth in internet services is becoming a business and social trend in many applications. It integrates the existing hardware resources to form a shared resource pool, making business systems obtain computing, storage, and network resources on demand, effectively solving the problems existing in traditional IT infrastructure [1–4]. However, due to the virtual and dynamic properties, it needs to do the shared resource allocation in a multi-tenant running environment, any inefficient management of resources may lead to serious social problems. Virtual machine (VM), as the core component of cloud platform, is responsible for providing computing and storage resources for business system to ensure the normal operations of the tasks. Therefore, under the current trends of large-scale platforms for sharing network resources, the VM abnormal behavior-oriented detection on the cloud computing is particularly important [5–8].

Anomaly detection refers to the process of discovering patterns in data that are inconsistent with the expected behavior, and the patterns that do not conform to the expected behavior are usually called exceptions or anomalies [9]. The basic idea of anomaly detection is to define one or several normal behavior patterns, and then detect whether the data contains patterns that are inconsistent with the normal behavior patterns. Based on this idea, several researches have been conducted to do the VM abnormal behavior detection on cloud platform and have achieved many results. According to the different intentions, the anomaly detection methods can be classified into several types, such as classification [10], outlier detection [11], regression [12], clustering analysis [13] and so on.

Nonetheless, anomalies are not always easy to detect. A general idea of abnormal behavior detection mechanism of VMs on cloud platform is to judge whether the VM in cloud platform has abnormal behavior with the help of

✉ Hancui Zhang
zhc_813@126.com

Weida Zhou
vindaz@163.com

[1] School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China

Published online: 17 August 2021

collecting and analyzing the performance index measurement values of VM on cloud platform. However, there are usually three problems in these kinds of methods: (1) many anomaly detection methods depend on static and labeled data, they ignore the dynamic and complexity nature of the cloud platform. Obviously, if the dataset is static, then the detection model is frozen. It is impossible to achieve a good detection result. (2) There are many performance indicators of VMs, the collected virtual machine performance index measurement value will constitute a high-dimensional virtual machine performance index space, which will increase the computational complexity of anomaly detection and degrade the performance of anomaly detection. (3) Currently, many virtual machines abnormal behavior detection algorithms only consider the performance index measurement value of VM itself and ignore the context information, then any significant change in the virtual machine performance index will be detected as an exception.

For these problems, in order to improve the accuracy of anomaly detection and reduce the computational complexity, this paper proposed an incremental virtual machine workload clustering algorithm, which clusters the virtual machine running state information with similar virtual machine workload information into the same cluster, and then adopted the incremental local outlier factor algorithm to meet the online anomaly detection needs, detecting some newly VM's running status on the basis of instant running state information. Based on this detection mechanism, it can effectively reduce the amount of data required for detection model training and detecting, so as to improve the efficiency and accuracy of detection. The main contributions of this paper are summarized as follows:

- In order to avoid the expensive detection cost and the low detection accuracy for the dynamic cloud platform, a two-stage virtual machine abnormal behavior-based detection mechanism is proposed, consisting of workload-based incremental clustering algorithm and the incremental local outlier factor algorithm.
- Integration of VM's workload and VM's performance indicators to optimize the detection model, which helps reduce data deviations; meanwhile, selection of the influenced VMs, not all of them, can facilitate the detection efficiency.
- Additive factors (workloads, detection times and accuracy) are incorporated to help evaluate the accuracy and efficiency of the proposed detection mechanism in the real-time private cloud platform.

The rest of this paper is organized as follows. Some related works are introduced in Sect. 2. A virtual machine abnormal behavior-based anomaly detection mechanism is presented in Sect. 3. Then, the theories and methodologies are specified in Sect. 4. Experiments and results are discussed in Sects. 5 and 6 concludes the paper.

2 Related work

In this section, the remarkable characteristics of the existing researches on both anomaly detection and the works on cloud computing are discussed.

2.1 Anomaly detection

Anomaly detection is a core element for service quality and network security by detecting unknown behaviors or attacks based on some profiles that defining the normal patterns. One of the main types of anomaly detection is classification, a supervised learning method to distinguish the anomalies by the model which is trained by priori knowledge. Besides, for some unknown data, which can not tell normal or abnormal directly, some unsupervised learning methods are presented for anomaly detection, such as clustering, dimension reduction methods [14–16], neural network [17, 18] and so on.

Gu et al. [19] proposed a modified support vector ordinal regression (SVOR) method and extended the accuracy of on-line v-support vector classification (v-SVC) algorithm to improve the detection accuracy under equality and inequality constraints.

Injadat et al. [20] tried to combine Bayesian optimization technique with Gaussian kernel-based support vector machine, Random Forest, and k-nearest neighbor algorithms to evaluate the detection efficiency. Another main type is clustering, an unsupervised learning method, which divides data or behaviors into different clusters based on the similarity between them.

Chen et al. [21] used clustering method to analyze the content of research articles and used graph social clustering analysis to visualize differences between authors. Yin and Zhang [13] combined improved DD algorithm with information entropy to tackle the sensitivity of traditional k-means to initial clustering centers. Then adopt the parallel processing to improve the detection efficiency. Besides, in recent years, with the boom of deep learning and network technology, the application of deep learning and data mining technique in anomaly detection has greatly improved the accuracy and efficiency of detection, drawing more and more public attention.

Ji et al. [22] focused on salient object detection (SOD) by leveraging deep CNN-based encoder–decoder model and presented an extensive empirical study. Meanwhile, the newly model is further evaluated on three video-based SOD benchmark datasets and demonstrated a good result. Ullah et al. [23] presented an efficient deep learning-based

anomaly detection framework. It first extracted features by a pre-trained convolutional neural network (CNN) model, then used a multi-layer bi-directional long short-term memory (BD-LSTM) model to classify ongoing anomalous or normal events in complex surveillance networks

2.2 Detection on cloud platform

Since the reliability of cloud platform have a serious impact on both society and individuals, various works are conducted by applying different methods to detect abnormal behaviors on cloud platform to ensure the dependable cloud environment.

Kc and Gu [24] designed a log-based fault checking system for large-scale cloud platforms, called efficient log-based troubleshooting system (ELT for short). The system can automatically extract key log information and perform invariant checking, greatly simplifying the fault checking task of system administrators.

Wang et al. [25, 26] proposed an entropy-based anomaly detection method for cloud platform anomaly detection. The experimental results show that, compared with the method based on threshold, the accuracy rate of anomaly detection using entropy-based anomaly detection method is improved by 57.4%; compared with the method based on approximate optimal threshold, the false alarm rate of anomaly detection method based on entropy is reduced by 59.3%.

Bhaduri et al. [27] designed an automatic fault detection framework based on distributed monitoring system ganglia for cloud platform. This fault detection framework not only has small additional cost, but also can effectively identify abnormal physical nodes in cloud platform.

Lopez [28] focused on DDoS attacks on the cloud platform, proposed a machine learning-based detection mechanism to learn the normal pattern of network traffic and the behavior of the network protocols to characterize the network behavior without consider the internal situation.

Zhang et al. [29] presented an anomaly detection system called PerfInsight, which combines with unsupervised clustering-based anomaly detection method with an entropy-based feature selection method to train some robust clustering models to improve detection efficiency.

Liu et al. [30, 31] designed a detection framework on the basis of self-organizing maps method to improving the efficiency for the data collection and detection, applied which an SOM network model was trained by the running properties of virtual machines. Then proposed a feature selection algorithm-based configuration parameter tuning method to improve the performance of the MapReduce model [32].

Besides the detection accuracy, the effectiveness of detection time is also attracted much attention. Alnafessa and Casale [33] combined neural network technology with anomaly detection method to deal with the in-memory running environment. Under the consideration of the trade-offs on precision and recall of the classifiers, it used a neural network model to check through the Spark logs data and a variety of monitoring metrics to do anomaly detection. Jindal et al. [34] focused on the memory leak problem to do the anomaly detection. It introduced a novel machine learning based algorithm, called Precog to detect the memory leak. It consists of two phases: online training and online detection. At the first stage, do collecting and pre-processing and it can be conducted routinely. Then in the second stage, a trend lines fitting module is used to do detection.

3 Virtual machine abnormal behavior-based detection mechanism

The proposed virtual machine abnormal behavior-based detection mechanism consists of two stages as shown in Fig. 1. In the first stage, the workload-based incremental clustering algorithm is used to cluster the VMs that have the similar running state information. Then when do clustering, the incremental local outlier factor algorithm is cooperated to update the relative information between the new VM running status information and VMs that is influenced by the new VM. With the help of control limit value of local outlier factor (LOF), it can detect the anomaly running status effectively.

This detection mechanism can deal with both offline and online detection needs. Obviously, in order to do the online detection precisely, it first needs relatively sufficient data samples to build a reliable offline model, which is called model training, then do detection and adjust the model to match the online requirement. The key point here for the online detection is that when do the adjustment of model or detection there is no need to handle all the VMs' running status, it only focus on the local range according to the variables like $\mathrm{MinPts}_{Distance}$, $\mathrm{MinPts}_{NNeighbor}$, $\mathrm{Reach}_{Distance}$, $local\_reach\_density$ and the $local\_Outlier\_Factor$. All these variables will be introduced in next section.

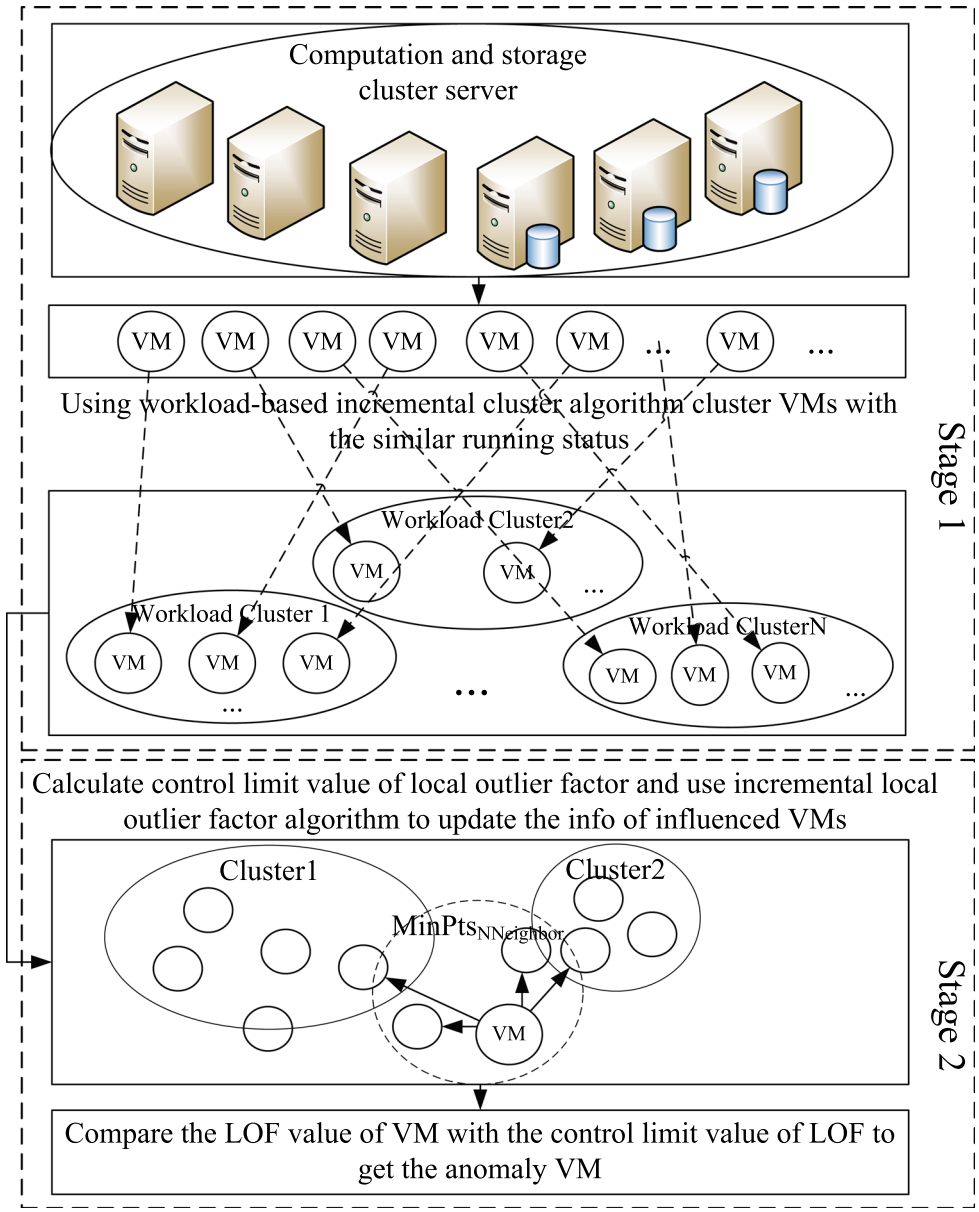

Fig. 1 The flowchart of the VM abnormal behavior-based detection mechanism

4 Theory and methodology

4.1 Virtual machine workload based incremental clustering method

Since the cloud platform is a continuous running production environment, it is necessary to continuously collect the running status information of VMs on the cloud platform, such as CPU, memory, storage, network bandwidth and so on, so as to realize the 24-h monitoring of the running status of VMs on the cloud platform [35]. At the same time, we observe that under different virtual machine workload, virtual machine performance indicators differ in measurement values, while under a similar virtual machine workload, the measurement values appear almost the same. Therefore, it is useful to take virtual machine workload into consideration for anomaly detection based on incremental clustering algorithm. Its purpose is to cluster the running state information of virtual machines with similar workload into the same cluster to avoid the interference of normal fluctuation under different loads.

Usually, in the cloud platform, each client VM with different customer requirements is scheduled by the VMM (virtual machine monitor) to coordinate operation on different physical servers. The performance of each client VM is affected by many metrics, such as the status of PM (physical machine) and customer applications (or demands). When it is running or dealing with applications (supporting services), all the hardware or software requirements can be described as the number of processes,

the number of sessions, concurrency level, network throughput and so on.

Suppose that the virtual machine workload vector at time t is expressed as $l_t = [l_{1t}, l_{2t}, \ldots, l_{it}, \ldots, l_{mt}]$, where $l_{it}$is the ith variable describing the workload of VM, which can be the number of processes, the number of threads, the number of sessions and so on. Similarly, the performance index of virtual machine at time t can be expressed as $p_t = [p_{1t}, \ldots, p_{it}, \ldots, p_{Dt}]$, where p_{it} is the ith variable of the performance index for the description of VM, which can be the computing resources, memory resources, network resources and so on. Therefore, the whole running status of VM at time t can be described as $R_t = (l_t, x_t)$, the dimension is $m + D$, and $m \ll D$, where m and D are the number of parameters that describe the VM workload and the performance index of the VM, separately.

Then, by reducing the performance index dimension, the running status vector of VM can be represented as $R_t = (l_t, \overline{p_t})$, where $\overline{p_t} = W^T p_t$, W is the d-dimensional vector to map the high D-dimensional performance index space data into low d-dimensional space. Besides, in order to measure the similarity of workload (or running state), the Mahalanobis distance is adopted to consider the relationships between variables. The distance formula is shown as below:

$$d_{ij} = \sqrt{(l_i - C_j)A^{-1}(l_i - C_j)^T}, \tag{1}$$

where C_j represents the workload vector of the center of jth cluster, that is,

$$C_j = \frac{1}{n}\sum_{k=1}^{n} l_k^j. \tag{2}$$

l_k^j displays the workload vector of the kth VM in cluster j. n is the size of cluster, that is the number of running VMs in the cluster.

Let A denotes the covariance matrix of the workload matrices composed of all the VMs in the cluster to show the correlation between the variables in each dimension. It can be formulated as:

$$A = \begin{bmatrix} COV(a_1, a_1) & \cdots & COV(a_1, a_m) \\ COV(a_2, a_1) & \cdots & COV(a_2, a_m) \\ \vdots & \ddots & \vdots \\ COV(a_m, a_1) & \cdots & COV(a_m, a_m) \end{bmatrix}, \tag{3}$$

where $a_i\,(1 \le i \le m)$ is the vector of ith workload variable. So the correlation between two variables (a_i and a_j) of the workload vector can be formulated as:

$$COV(a_i, a_j) = \frac{1}{n}\sum_{k=1}^{n}(a_{ik} - \overline{a_i})(a_{jk} - \overline{a_j}), \tag{4}$$

where $\overline{a_i}$ is the average value of ith workload variable, that is, $\overline{a_i} = \frac{1}{n}\sum_{k=1}^{n} a_{ik}$.

In Fig. 2, the Algorithm 1 presents the pseudocode of the virtual machine workload-based incremental clustering method. It takes the current running status vector of the VMs as input, due to the running state changes along with the time, so the whole state can be expressed in the form of a state sequence $R = \{R_1, R_2, \ldots, R_t, \ldots\}$, which consists of two parts, workload l_t and performance $\overline{p_t}$. Then try to find the most similar VMs to gather into a cluster with the consideration of the VM workload. If there is no cluster exist at that time, create a new cluster with only the current running status info of the VM using the function $CreateCluster(l_t)$. When execute the creation function the main factor is workload l_t. Otherwise, if there exists some clusters, then do the similarity calculation based on the Mahalanobis distance. Choose the cluster with the minimum distance $cluster\_index_{l_t}$ as the candidate cluster. Here, in order to keep the quality of clusters, a distance constant $dist_{threshold}$ is adopted. If the minimum distance between l_t and the candidate cluster less than the distance constant $dist_{threshold}$, then add it into the candidate cluster $AddIntoCluster(R_t, cluster\_index_{l_t})$ and update the cluster $UpdateCenter(cluster_{index_{l_t}})$. Otherwise, create a new cluster $CreateCluster(l_t)$ on the basis of the workload l_t. After all the clusters have been calculated, terminate the while loop and get the workload clusters.

4.2 Incremental local outlier factor method

The local outlier factor algorithm (LOF) was first proposed by Breunig et al. [36] of University of Munich, Germany. It is a density-based anomaly detection method. The main idea of it is to determine whether the pattern or behavior data object is distributed in a relatively concentrated local area by giving each pattern or behavior data object a factor to represent the abnormal degree, which has been widely used in anomaly detection on cloud platform.

However, due to the dynamic characteristics of cloud platform, its data set is dynamic, too. When the running state information of VMs R_t in the cluster changes, the original local outlier factor algorithm usually needs to recalculate the value of all VMs' running state information in the cluster, including the factors $MinPts_{Distance}$, $MinPts_{NNeighbor}$, $Reach_{Distance}$, $local\_reach\_density$ and $local\_Outlier\_Factor$, resulting in large amount of calculation and high time complexity [37]. Moreover, it is found that inserting a new VM's running state info R_t into the cluster or deleting one old info from it usually only affects some VMs' states, there is no need to recalculate all the VMs' running state info.

Fig. 2 The pseudo code of the VM workload-based incremental clustering algorithm

Algorithm 1: Virtual Machine Workload-Based Incremental Clustering Algorithm

Input:

A sequence of running status of VM $R = \{R_1, R_2, \ldots, R_t, \ldots\}$, where $R_t = (l_t, \overline{p}_t)$ denotes the running information of VM at time t, l_t is the workload vector of VM at time t and $\overline{p}_t$ is the performance index vector.

Set distance threshold of a cluster to a constant called $dist_{threshold}$. Define a loop flag named $Flag$ and initialize the value to $TRUE$.

Output:

Get the Cluster of VM with the consideration of workload and performance index.

1. While(Flag)
2. GetFlag(Flag); // get a flag to exit the while loop
3. GetRunningStatus();// get $R_t = (l_t, \overline{p}_t)$
4. IF there is no workload cluster exist
5. THEN $CreateCluster(l_t)$;
6. CONTINUE;
7. ELSE
8. FOR each cluster j
9. $d_{tj} = \sqrt{(l_t - C_j)A^{-1}(l_t - C_j)^T}$
10. $cluster\_index_{l_t} = index(\min_{1 \le j \le CS} d_{tj})$
11. IF $dist\left(l_t, cluster_{index_{l_t}}\right) > dist_{threshold}$
12. THEN $CreateCluster(l_t)$;
13. CONTINUE;
14. ELSE
15. AddIntoCluster($R_t, cluster\_index_{l_t}$);
16. UpdateCenter($cluster\_index_{l_t}$);

Therefore, an incremental local outlier factor algorithm (ILOF) is proposed to reduce the computational cost. The insertion and deletion processes are demonstrated in Algorithm 2 (as shown in Fig. 3). The core is to judge what the process is and then choose the corresponding operations. That is, if the current process is inserting a new info R_t, it needs to do $AddIntoCluster(R_t, j)$ first, and calculate the $MinPts_{NNeighbor}(R_t)$ after the insertion. Then any VMs in the range of $MinPts_{NNeighbor}(R_t)$ needs to be updated. Else if the current process is to delete an info R_t, it should record the $MinPts_{NNeighbor}(R_t)$ first then do the deletion process. After that, whether it is insertion or deletion, VMs in the range of the reachability distance from the target R_t need to be updated, including $MinPts_{Distance}$, $MinPts_{NNeighbor}$, $Reach_{Distance}$, $local\_reach\_density$ and $local\_Outlier\_Factor$, as demonstrated in line 9-23. Finally, for insertion, it needs to update the local reach density $Update_{lrd}(R_t)$ and the local outlier factor $Update_{lof}(R_t)$ of the new R_t, while for deletion, do nothing.

4.3 Kernel density estimation-based control limit calculation method

From above, it is easy to find that in order to achieve the purpose of anomaly detection, it is necessary to calculate the control limit of local outlier factors of some influenced normal data samples. The traditional calculation methods of control limits assume that normal data samples are independent of each other and obey normal distribution. However, in a complex real cloud environment, the distribution of VM running status cannot be reduced to a simple distribution model [38–40]. In order to avoid the assumption that normal data samples obey normal distribution. A data-driven kernel density estimation method is

Fig. 3 The pseudo code of the insertion and deletion process

Algorithm 2: The Insertion or Deleting process of R_t

Input:

A target cluster j, which has the similar workload with the new running information of VM at time t, the running state info $R_t = (l_t, \overline{p}_t)$, a threshold parameter $MinPts$.

Output:

None.

1. IF insert R_t Then
2. $AddIntoCluster(R_t, j)$
3. $MinPts_{NNeighbor}(R_t) = \{q \in D - \{R_t\} | d_{tq} \leq MinPts_{Distance}(R_t)\}$
4. FOR any $q \in MinPts_{NNeighbor}(R_t)$
5. $Update_{Reach_{Distance}}(R_t, q)$;
6. Else if delete R_t Then
7. $MinPts_{NNeighbor}(R_t) = \{q \in D - \{R_t\} | d_{tq} \leq MinPts_{Distance}(R_t)\}$
8. $DeleteFromCluster(R_t, j)$
9. $I_{MinPts_{Distance}}(R_t) = GetInfluenceVM_{Dist}(R_t)$
10. FOR any $q \in I_{MinPts_{Distance}}(R_t)$
11. $Update_{MinPts_{Distance}}(q)$
12. $I_{MinPts_{NNeighbor}}(R_t) = GetInfluenceVM_{NNeighbor}(R_t)$
13. FOR any $q \in I_{MinPts_{NNeighbo}}(R_t)$
14. $Update_{MinPts_{NNeighbor}}(q)$
15. $I_{Reach_{Distance}}(R_t) = GetInfluenceVM_{ReachDist}(R_t)$
16. FOR any $q \in I_{Reach_{Distance}}(R_t)$
17. $Update_{Reach_{Distance}}\left(q, MinPts_{NNeighbor}(q)\right)$
18. $I_{lrd}(R_t) = GetInfluenceVM_{localRD}(R_t)$
19. FOR any $q \in I_{lrd}(R_t)$
20. $Update_{lrd}(q)$
21. $I_{lof}(R_t) = GetInfluenceVM_{localOF}(R_t)$
22. FOR any $q \in I_{lof}(R_t)$
23. $Update_{lof}(q)$
24. IF $AddIntoCluster(R_t, j)$ is TRUE, then
25. $Update_{lrd}(R_t)$
26. $Update_{lof}(R_t)$

proposed to calculate the control limits of local outlier factors among the influenced normal VM running status data.

Suppose there is a sequence of local outlier factors of some influenced normal VM running status information $\{lof_1, lof_2, \ldots, lof_i, \ldots\}$, then the probability density function of local outlier factor variable can be defined as follows:

$$P(lof) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{lof - lof_i}{h}\right), \tag{5}$$

where h is the smoothing parameter, n is the number of data samples, $K(\cdot)$ is the kernel function and Gaussian kernel is selected, that is, $K(\mu) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\mu^2}$, where $\mu = \frac{lof - lof_i}{h}$.

As can be seen from Eq. (5), when the number of running state samples n is sufficient, the accuracy of kernel density estimation function entirely depends on the selection of smoothing parameter h and kernel function $K(\cdot)$. And some previous studies have found that when h is optimal, the influence of different kernel functions on the accuracy of kernel density estimation function is almost equivalent, while when $K(\cdot)$ is fixed, the influence of different smoothing parameter on the accuracy of kernel density estimation function is great [41]. Besides, if h is too

large, then the probability density estimation function displays too smooth, leading to ignore some important features; else if h is set to be too small, then it will present large fluctuation on the probability density estimation function. So, how to choose the suitable h is the key to kernel-based density estimation.

In this paper, an asymptotic mean integrated squared error method (AMISEM) is adopted to calculate the smoothing parameter h. It can be formulated as:

$$AMISEM(P(lof)) = \frac{I(K)}{nh} + \frac{h^4}{4} I(P'')I(\mu), \tag{6}$$

where $I(K) = \int K^2(\mu)d\mu$, $I(P'') = \int [P''(lof)]^2 d(lof)$, $P''(lof)$ is the second derivative of possibility density estimation function, $I(\mu) = \int \mu^2 K(\mu)d\mu$.

The goal of Eq. (6) is to get the minimum asymptotic mean integrated squared error, so do the partial derivative around h:

$$\frac{\partial AMISEM(P(lof))}{\partial h} = \frac{I(K)}{nh^2} + h^3 I(P'')I(\mu) = 0. \tag{7}$$

Then, get the optimal smoothing parameter h:

$$h_{opt} = \left[\frac{I(K)}{I(\mu)I(P'')}\right]^{\frac{1}{5}} n^{\frac{-1}{5}}. \tag{8}$$

It can be seen from Eq. (8) that h depends on $I(P'')$ as the value of $K(\cdot)$ and μ are already known. From Eqs. (6) and (5), $I(P'')$ is related to $P(lof)$, so this paper applies the Rule of Thumb to use Gaussian kernel function $K(\cdot)$ as approximate value of $P(lof)$. Then Eq. (8) can be rewritten as:

$$h_{opt} = \left[\frac{I(K)}{I(\mu)I(P'')}\right]^{\frac{1}{5}} n^{\frac{-1}{5}} = \left[\frac{(2\sqrt{\pi})^{-1}}{\frac{3}{8}\pi^{\frac{-1}{2}}\sigma^{-5}}\right] n^{\frac{-1}{5}} = \left[\frac{4\sigma^5}{3n}\right]^{\frac{1}{5}}, \tag{9}$$

where σ is the standard deviation.

Combine with Eqs. (9) and (5), it is easy to get the possibility density value of the local outlier factor $P(lof)$. After this, the control limit of the local outlier factor $lof(\alpha)$ can be calculated by using the formula:

$$\int_{lof(\alpha)}^{\infty} P(lof)d(lof) = \alpha, \tag{10}$$

where α is the significance factor, here, we set it to be 1%.

5 Experiments and results

5.1 Experimental environment

In order to test the proposed detection mechanism for cloud platform, six physical servers are collected as a cloud testbed. One of them is set to be the controller, the others are selected as the compute nodes and each allocated at most four VMs. The tools used to manage this private cloud platform is Openstack, and the virtualization tool is Xen 3.2.0. The experimental environment is shown in Fig. 4. On the control node runs control unit to manage the platform, while on each computing node runs detectors and monitor units.

In order to verify the effectiveness of the proposed detection mechanism based on the workload-based incremental clustering algorithm and the incremental local anomaly factor algorithm on line, a distributed online service benchmark RUBiS [42, 43] is introduced. It is deployed to the built cloud computing test bed, and then randomly injects faults related to CPU, memory, disk, and network into the monitored virtual machine by fault injection program [44–46], including HTTP request, application logic operations and data storage and management and so on.

5.2 Experimental results and analysis

5.2.1 Experiment 1: the impact of virtual machine workload-based clustering algorithm on anomaly detection performance.

In this experiment, two online detection methods are conducted to check the impact of VM workload on anomaly detection as shown in Fig. 5. One uses the full framework of the mechanism (with workload-based incremental clustering algorithm, WITH) and another only has stage 2 of the mechanism (without workload-based incremental clustering algorithm, WITHOUT). Five different workloads are set: 10%, 30%, 50%, 70%, 90% and each is executed 10 times. Each time 50–90 faults are injected into the VM randomly. Log the VM running performance index and the detection results. The average results are shown in Figs. 5 and 6.

From Fig. 5, it can be found that in the simulated highly dynamic cloud platform, the detection mechanism with virtual machine workload-based incremental clustering algorithm (WITH) proposed in this paper can effectively maintain the accuracy of anomaly detection. As in a highly dynamic cloud platform, due to the continuous change of virtual machine workloads, the performance index value of virtual machine fluctuates normally. The anomaly detection mechanism without virtual machine workload-based clustering algorithm (WITHOUT) appears misjudgment, taking the normal fluctuation as abnormal, which leads to the decrease of true positive rate; while using the virtual machine workload-based incremental clustering algorithm (WITH), it analyzes the running state information first, gathering the virtual machines with similar workload into cluster, which can build a better view of the whole

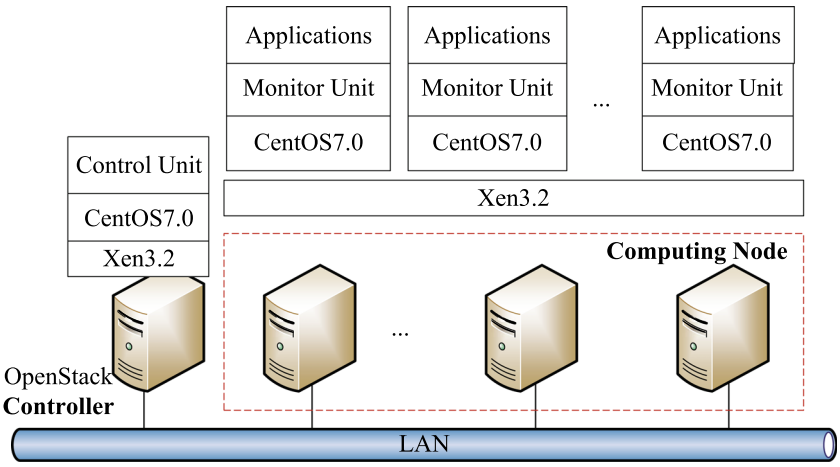

Fig. 4 The experimental environment

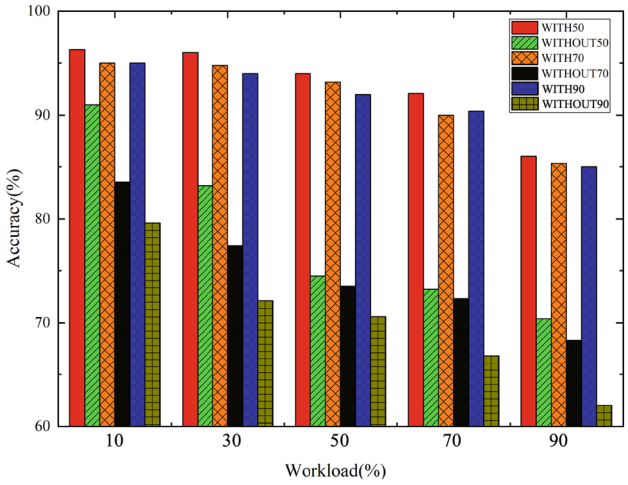

Fig. 5 The Accuracy for anomaly detection using WITH and WITHOUT workload-based incremental clustering algorithm within five different workloads (10%, 30%, 50%, 70%, 90%) on three different number of faults (50, 70, 90)

environment to improve the detection accuracy. After that, do the detection based on incremental local outlier factor algorithm introduced in Sect. 4.

Figure 6 displays the time complexity and accuracy between WITH and WITHOUT methods as the number of injected faults is fixed to be 50. As the workload changing, it is easy to find that both methods have the same trends of accuracy (descending) and the time cost (increasing). But when the workload is 35%, the WITHOUT method has a break-point that the detection times grows fast and the accuracy has downed to 75%, while WITH method has a relatively smooth growing in time and a reliability accuracy. That means this WITH method can better deal with the complex cloud environment with the changing workload.

5.2.2 Experiment 2: performance comparison between two online anomaly detection mechanisms.

In this experiment, the original local outlier factor algorithm (OLOF) is used to compare with the proposed incremental local outlier factor algorithm (ILOF) to verify the impact of the efficiency of the detection mechanism. 20 VMs are initialized first with the same workload and 50 faults are intended to be randomly injected into 1 VM during the experiment as well as the application requests, the number of which will in the range of [500, 2000], including the computational requests, storage requests, http requests and so on, which are used to simulated as the changing of VM's running state. 10-Fold experiments are executed, the average results are shown in Figs. 7 and 8.

Figure 7 illustrates the detection accuracy of OLOF and ILOF algorithm when the virtual machine running state info changes. In the figure, horizontal axis denotes the interval of time to produce users' requests to simulate the changing of VMs and the vertical axis is the average detection accuracy of each case. Here, five different time intervals are used to analyze the performance under four different user requests that is R500, R1000, R1500 and R2000. The results shown in Fig. 7 indicate that both OLOF and ILOF can handle the dynamic environment efficiently and ILOF has a slightly higher detection accuracy.

Figure 8 displays the efficiency between OLOF and ILOF on the online anomaly detection. It can be seen from the results that the incremental local outlier factor (ILOF) algorithm proposed in this paper greatly reduces the average computing time cost, improving the real-time performance of online anomaly detection. As shown in Algorithm 2, when inserting a new virtual machine running state information or deleting a virtual machine running state information from the cluster, the original local outlier factor algorithm (OLOF) needs to update the *lof* value by

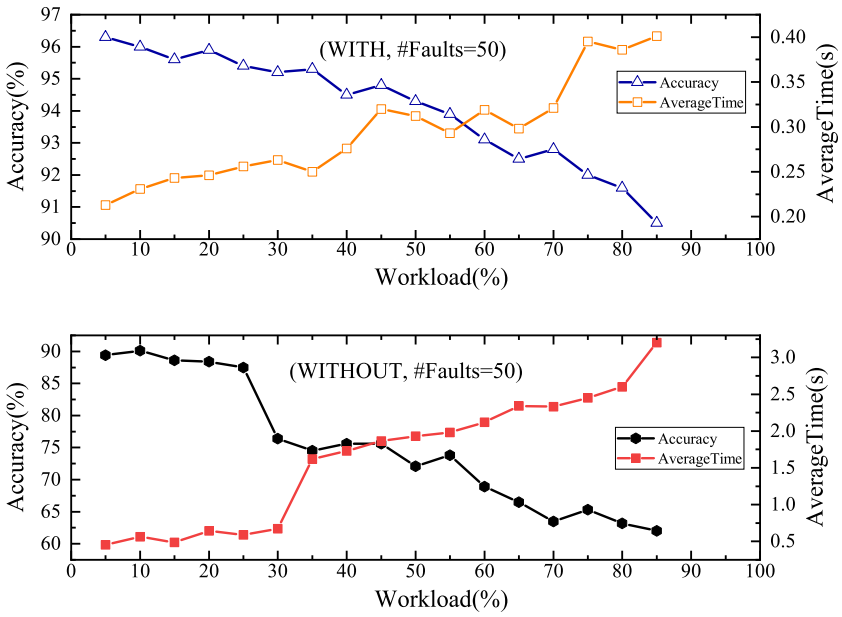

Fig. 6 Comparison of the accuracy and efficiency between WITH and WITHOUT algorithm on 50 faults as the workload changes

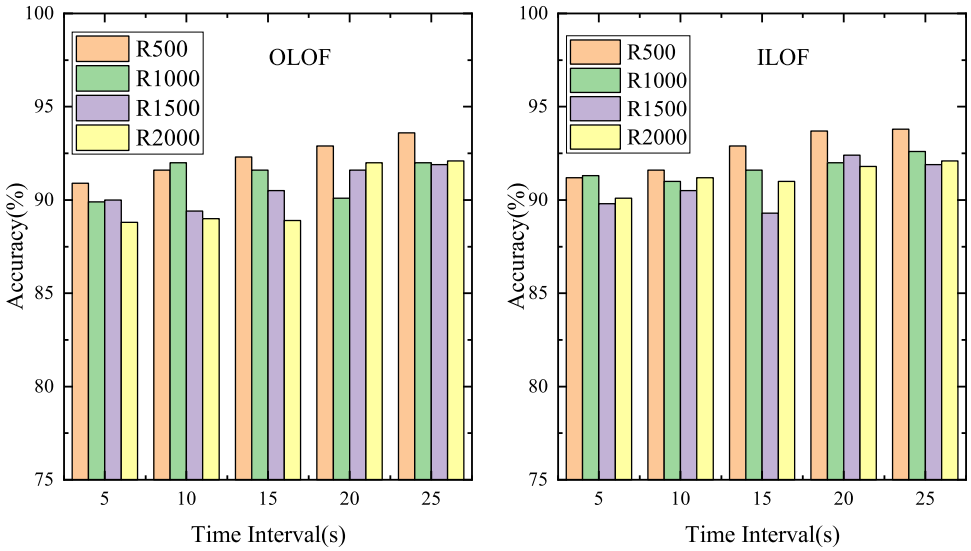

Fig. 7 Comparison of the accuracy between OLOF and ILOF algorithm within four running environments (R500, R1000, R1500, R2000). In each environment the time interval to produce the application requests is set to be 5, 10, 15, 20, and 25 s

recalculating all virtual machine running state information in the cluster, while the incremental local outlier factor algorithm (ILOF) proposed in this paper only needs to update the *lof* value by the affected virtual machine running state information, which is much smaller. Therefore, for the online anomaly detection on the dynamic cloud platform, ILOF works better than OLOF.

6 Conclusion

By studying the relationship between virtual machine workload and virtual machine performance indicators, the combination of these two information will significantly reduce the false positive rate of virtual machine anomaly detection. At the same time, the incremental local outlier factor algorithm is proposed to reduce the computational complexity by updating only the local outlier values of the affected virtual machines in the cluster. In addition, the control limit calculation method of kernel density estimation to calculate the local outlier factor value of normal virtual machine running state information in the cluster is also designed for the dynamic cloud platform.

Two experiments are conducted to verify the accuracy and efficiency of this detection mechanism and the results show that the combination of the workload-based incremental clustering method and the incremental local outlier factor algorithm can not only improve the accuracy of the anomaly detection, but also greatly reduce the average computing time cost. In the future, it would be interesting

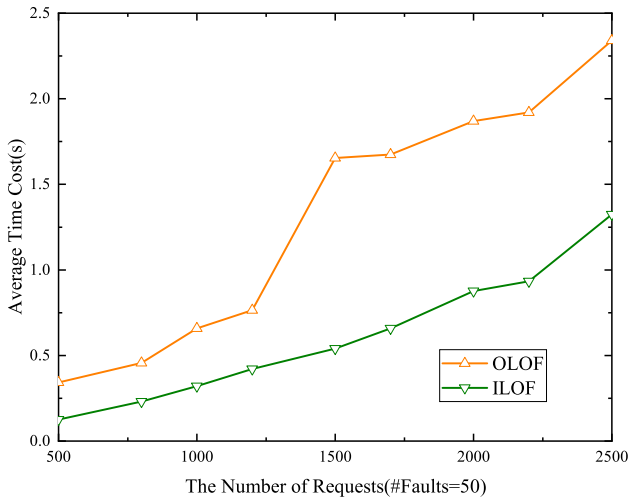

Fig. 8 Efficiency comparison between OLOF and ILOF algorithm for the anomaly detection

to explore the demand-workload model to optimize the workload-based incremental clustering algorithm. Meanwhile, deep learning techniques will be an effective method to learn more robust models to meet the challenges of the complex cloud platform [47–50].

References

1. Chiba, Z., Abghour, N., Moussaid, K., et al.: A survey of intrusion detection systems for cloud computing environment. In: Presented at the 2016 International Conference on Engineering and MIS (ICEMIS) (2016)
2. Razaque, A., Amsaad, F., Hariri, S., et al.: Enhanced Grey risk assessment model for support of cloud service provider. IEEE Access **99**, 1 (2020)
3. Jian, Z., Zheng, L., Gong, L., et al.: A survey on security of cloud environment: threats, solutions, and innovation. In: 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC). IEEE Computer Society (2018)
4. Daniya, T., Suresh Kumar, K., Santhosh Kumar, B., Chandra Sekhar, K.: A survey on anomaly based intrusion detection system. Mater. Today Proc. (2021). https://doi.org/10.1016/j.matpr.2021.03.353
5. Fargo, F., Franza, O., Tunc, C., et al.: Autonomic resource management for power, performance, and security in cloud environment. In: 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA). IEEE (2019)
6. Doshi, R., Kute, V.: A review paper on security concerns in cloud computing and proposed security models. In: 2020 International Conference on Emerging Trends in Information Technology and Engineering (IC-ETITE). (2020)
7. Pan, Z., Hariri, S., Pacheco, J.: Context aware intrusion detection for building automation systems. Comput. Secur. **85**(Aug.), 181–201 (2019)
8. Kumbhare, N., Marathe, A., Akoglu, A., et al.: A value-oriented job scheduling approach for power-constrained and oversubscribed HPC systems. IEEE Trans. Parallel Distrib. Syst. **31**(6), 1419–1433 (2020)
9. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. **41**(3), Article No. 15 (2009)
10. Gu, B., Sheng, V.S.: A robust regularization path algorithm for ν-support vector classification. IEEE Trans. Neural Netw. Learn. Syst. **28**(5), 1241–1248 (2017)
11. Kumar, M., Mathur, R.: Unsupervised outlier detection technique for intrusion detection in cloud computing. In: IEEE International Conference for Convergence of Technology (I2CT). IEEE (2015)
12. Song, C., Wang, Y., Zhang, S., et al.: A low cost and easy implement highway accident detection model based on Big Data. In: 2019 IEEE International Conferences on Ubiquitous Computing and Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (Smart CNS). IEEE (2020)
13. Yin, C., Zhang, S.: Parallel implementing improved k-means applied for image retrieval and anomaly detection. Multimed. Tools Appl. **76**(16), 16911–16927 (2017)
14. Adler, N., Yazhemsky, E.: Improving discrimination in data envelopment analysis: PCA-DEA or variable reduction. Eur. J. Oper. Res. **202**(1), 273–284 (2010)
15. Koziel, S., Pietrenkodabrowska, A.: Low-cost data-driven modelling of microwave components using domain confinement and PCA-based dimensionality reduction. IET Microw. Antennas Propag. **14**(13), 1643–1650 (2020)
16. Li, Z., Yan, X.: Fault-relevant optimal ensemble ICA model for non-Gaussian process monitoring. IEEE Trans. Control Syst. Technol. **28**(6), 2581–2590 (2020)
17. Advani, M.S., Saxe, A.M.: High-dimensional dynamics of generalization error in neural networks. Neural Netw. **132**, 428–446 (2020)
18. Stephanakis, I.M., Chochliouros, I.P., Sfakianakis, E., Shirazi, S.N.: Hybrid self-organizing feature map (SOM) for anomaly detection in cloud infrastructures using granular clustering based upon value-difference metrics. Inf. Sci. **494**(C), 247–277 (2019)
19. Gu, B., Sheng, V.S., Tay, K.Y., Romano, W., Li, S.: Incremental support vector learning for ordinal regression. IEEE Trans. Neural Netw. Learn. Syst. **26**(7), 1403–1416 (2017)
20. Injadat, M., Salo, F., Nassif, A.B., et al.: Bayesian optimization with machine learning algorithms towards anomaly detection. In: IEEE Global Communications Conference. IEEE (2018)
21. Chen, C.C., Fu, X., Chang, C.Y.: A terms mining and clustering technique for surveying network and content analysis of academic groups exploration. Clust. Comput. **20**(1), 43–52 (2017)
22. Ji, Y., Zhang, H., Zhang, Z., et al.: CNN-based encoder–decoder networks for salient object detection: a comprehensive review and recent advances. Inf. Sci. **546**, 835–857 (2021)
23. Ullah, W., Ullah, A., Haq, I.U., et al.: CNN features with bi-directional LSTM for real-time anomaly detection in surveillance networks. Multimed. Tools Appl. (2020). https://doi.org/10.1007/s11042-020-09406-3
24. Kc, K., Gu, X.: ELT: efficient log-based troubleshooting system for cloud computing infrastructures. In: Proceedings of the IEEE Symposium on Reliable Distributed Systems, pp. 11–20 (2011)
25. Wang, C., Schwan, K., Wolf M.: EbAT: an entropy based online Anomaly Tester for data center management. In: 2009 IFIP/IEEE International Symposium on Integrated Network Management-Workshops. IEEE (2009)
26. Wang, C., Talwar, V., Schwan, K., et al.: Online detection of utility cloud anomalies using metric distributions. In: IEEE/IFIP Network Operations and Management Symposium. IEEE (2010)
27. Bhaduri, K., Das, K., Matthews, B.L.: Detecting abnormal machine characteristics in cloud infrastructures. In: 2011 IEEE 11th International Conference on Data Mining Workshops (ICDMW), Vancouver, BC, Canada. IEEE (2012)
28. Lopez, A.D., et al.: Network traffic behavioral analytics for detection of DDoS attacks. SMU Data Sci. Rev. **2**(1), 14 (2019)

29. Zhang, X., Meng, F., Xu, J.: PerfInsight: a robust clustering-based abnormal behavior detection system for large-scale cloud. In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), pp. 896–899 (2018)
30. Liu, J., Chen, S., Zhou, Z., et al.: An anomaly detection algorithm of cloud platform based on self-organizing maps. Math. Probl. Eng. **4**, 1–9 (2016)
31. Liu, J., Zhang, H., Xu, G.: An anomaly detector deployment awareness detection framework based on multi-dimensional resources balancing in coud platform. IEEE Access **6**, 44927–44933 (2018)
32. Liu, J., Tang, S., Xu, G., et al.: A novel configuration tuning method based on feature selection for Hadoop MapReduce. IEEE Access **99**, 1 (2020)
33. Alnafessa, H.A., Casale, G.: Artificial neural networks based techniques for anomaly detection in Apache Spark. Clust. Comput. **23**(4), 1–16 (2020)
34. Jindal, A., Staab, P., Cardoso, J., et al.: Online memory leak detection in the cloud-based infrastructures. In: International Workshop on Artificial Intelligence for IT Operations (AIOPS) 2020 (2021)
35. Luo, J., Tang, J., Xiao, X.: Abnormal gait behavior detection for elderly based on enhanced Wigner–Ville analysis and cloud incremental SVM learning. J. Sens. **2016**, 1–18 (2016)
36. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, vol. 29(2), pp. 93–104 (2000)
37. Guizani, N., Ghafoor, A.: A network function virtualization system for detecting malware in large IoT based networks. IEEE J. Sel. Areas Commun. **99**, 1 (2020)
38. Alatawi, S., Alhasani, A., Alfaidi, S., et al.: A survey on cloud security issues and solution. In: 2020 International Conference on Computing and Information Technology (ICCIT-1441) (2020)
39. Hussain, S.A., Fatima, M., Saeed, A., et al.: Multilevel classification of security concerns in cloud computing. Appl. Comput. Inform. **13**(1), 57–65 (2017)
40. Liu, J., Zheng, S., Xu, G., et al.: Cross-domain sentiment aware word embeddings for review sentiment analysis. Int. J. Mach. Learn. Cybern. **12**(5), 1–12 (2020)
41. Coppolino, L., D'Antonio, S., Mazzeo, G., et al.: Cloud security: emerging threats and current solutions. Comput. Electr. Eng. **59**, 126–140 (2017)
42. Yazdanov, L., Fetzer, C.: VScaler: autonomic virtual machine scaling. In: IEEE Sixth International Conference on Cloud Computing. IEEE (2013)
43. Chen, T., Bahsoon, R.: Self-adaptive and online QoS modeling for cloud-based software services. IEEE Trans. Softw. Eng. (2017). https://doi.org/10.1109/TSE.2016.2608826
44. Hosseini, F.S., Yang, C.: Comprehensive evaluation of program reliability with ComFIDet: an integrated fault injection and detection framework for embedded systems. In: 2019 IEEE International Conference on Embedded Software and Systems (ICESS). IEEE (2019)
45. Alexandersson, R., Karlsson, J.: Fault injection-based assessment of aspect-oriented implementation of fault tolerance. Lect. Notes Comput. Sci. **6351**(4), 466–479 (2010)
46. Chen, X., Lin, J., Ma, Y., et al.: Self-adaptive resource allocation for cloud-based software services based on progressive QoS prediction model. Sci. China Inf. Sci. **62**(11), 1–3 (2019)
47. Sethi, K., Kumar, R., Prajapati, N., et al.: Deep reinforcement learning based intrusion detection system for cloud infrastructure. In: 2020 International Conference on COMmunication Systems and NETworkS (COMSNETS) (2020)
48. Ferrari, P., Rinaldi, S., Sisinni, E., et al.: Performance evaluation of full-cloud and edge-cloud architectures for industrial IoT anomaly detection based on deep learning. In: 2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0 & IoT) (2019)
49. Huang, J., Jiang, Z., Gong, L., et al.: Construction of hidden fault channel cloud test platform based on deep learning. In: 2020 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA) (2020)
50. Kimmel, J.C., Mcdole, A.D., Abdelsalam, M., et al.: Recurrent neural networks based online behavioural malware detection techniques for cloud infrastructure. IEEE Access **9**, 68066–68080 (2021)

Hancui Zhang received her Ph.D. Degree in School of Big Data and Software Engineering from Chongqing University, People's Republic of China, in June 2018. Currently, she is a Teacher in School of Information Science and Technology, Zhejiang Sci-Tech University, People's Republic of China. Her current interests include cloud computing, largescale data mining, flash memory, and fault detection.

Weida Zhou received his MA in School of Information Science and Technology from Zhejiang Sci-Tech University, People's Republic of China, in April 2006. Since 2006, he has been a Teacher in School of Information Science and Technology, Zhejiang Sci-Tech University, People's Republic of China. His current interests include computer vision, pattern recognition and artificial intelligence.