Received October 31, 2018, accepted November 25, 2018, date of publication December 3, 2018,
date of current version December 27, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2884508

Adaptive and Incremental-Clustering Anomaly Detection Algorithm for VMs Under Cloud Platform Runtime Environment

HANCUI ZHANG[1], JUN LIU[2], AND TIANSHU WU[3]
[1]School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China
[2]College of Software Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[3]College of Software Engineering, Chongqing University, Chongqing 400044, China

Corresponding author: Hancui Zhang (zhc_813@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572090, in part by the Science Foundation of Zhejiang Sci-Tech University under Grant 18032230Y, in part by the Chongqing Municipal Education Committee through the Science and Technology Research Project under Grant KJ1704081, and in part by the Chongqing Research Program of Basic Research and Frontier Technology under Grant cstc2017jcyjAX0270.

ABSTRACT The advent of cloud platform has promoted the complexity and scales of industries increasingly. Any deliberate or non-deliberate faults may cause enormous impact on system performance and server costs. Anomaly detection is a good way to identify anomalies and improve the dependability of the cloud platform. However, some of the anomaly detection methods are labeled data dependency, and some of them are sensitive to the dynamic runtime environment of the cloud platform. To address the problems, an adaptive and incremental clustering anomaly detection algorithm for virtual machines under the cloud platform runtime environment is proposed. Compared with the previous detection methods, the effect of the runtime environment factor is taken into account. Owning to the high level of dynamic cloud platform manages and the resources allocation of virtual machines, the environmental factors play an important role in the running performance of the virtual machines. In this paper, an improved adaptive and incremental clustering algorithm is introduced to perform the detection with the considerations of the cloud platform runtime environment. To demonstrate the effectiveness, two sets of experiments are performed. The experimental results indicate that the proposed anomaly detection method can greatly improve the detection accuracy rate even the cloud platform runtime environment changes.

INDEX TERMS Anomaly detection, runtime environment, cloud platform, incremental clustering, virtual machine.

I. INTRODUCTION

As the increasing of cloud platform scale and the growing number of features, cloud platform has become increasingly complex. The complexity as well as the inherent characteristics of cloud platform for sharing is easy to produce some performance anomalies that degrade performance, downtimes of virtual machines, lowing the dependability of cloud platform. These performance anomalies are usually caused by resource competition [1], [2], performance bottlenecks [3], error configuration [4], software detects [5], hardware failure [6] and external attacks [7] and so on. Anomaly detection is a good way could help to find and identify abnormal behaviors before real failures occur to improve the dependability of cloud platform.

Virtualization technology, especially the Host Virtualization Technology [8] has become one of the key supporting technologies to build a cloud data center in cloud platform. At present, most of the mainstream cloud service providers, such as IBM, Google, Microsoft, and Amazon and so on, have adopted this virtualization technology to provide their own cloud computing solutions and related technologies, which make the virtual machines as the research object of anomaly detection, become a trend in the cloud platform.

The anomalies of a virtual machine refer to any sudden performance degradation deviating from the normal behavior. Different from the collapse failure of virtual machine to stop sunning immediately, abnormal performance usually results in the decrease of virtual machine running efficiency. For instance, if the cloud server appears insufficient of memory and disk space and results in abnormal virtual machine performance, the user will find that the virtual machine can still run normally, in addition to the lower speed and longer response time. And research shows [9]–[11] that most of the abnormal performance usually comes with a significant change in system performance metrics, so it is a good way to detect virtual machine anomalies by collecting and analyzing of system running performance metrics.

However, due to the large scale of cloud platform and the large variety of operating systems in the virtual machines, continuously monitoring of all virtual machines leads to the overwhelming volume of performance metric data along with a high metric dimensionality. What's more, in the real-world cloud platform, the labeled data is almost not available. Therefore, due to the large number, high dimensionality and unlabeled performance metric data, most of the existing anomaly detection methods will not suitable for the needs of virtual machine anomaly detection. They are mostly underlying assumptions of the availability of training data set that has been labeled, such as SVM-based [12], neural networks-based [13], Bayesian network-based [14], [15], and statistic-based [16]. Besides, the acquisition of statistic model or classifier for each virtual machine of these detection methods will bring huge time overhead and computational cost.

In addition, the cloud platform is flexible enough to allow consumers to initiate and terminate virtual machines dynamically according to specific task requirements. How to effectively detect the abnormal performance of virtual machines on line and be adaptive to the high dynamic running environment is a challenging. Meanwhile, the running performance metric of each virtual machine will be affected by the running environment factors, such as the virtual machine resource configuration, the virtual machine system workload and the host resource allocation (shown in Figure 1). Therefore, if anomaly detection with the virtual machine runtime performance metric data alone, lacking of the awareness of the runtime environment, then any significant changes of the virtual machine performance metric will be regarded as anomalies, leading to a high false alarm rate and reducing the detection accuracy.

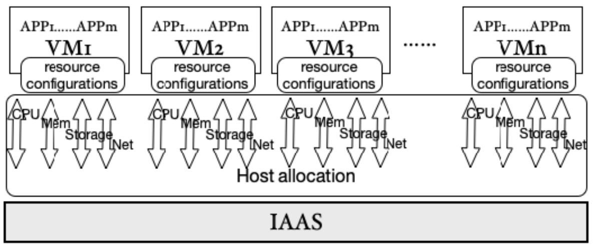


FIGURE 1. The runtime environment factors of virtual machines in cloud platform (IAAS).

In order to meet the dynamic characteristics of the large scale cloud platform, reduce false alarm rate and improve detection accuracy, this paper proposes an adaptive and incremental clustering anomaly detection algorithm for virtual machines under the cloud platform runtime environment. First, take the virtual machine anomaly detection as a contextual anomaly with the runtime environment as the context of virtual machines, underlying the assumption that similar running performance appears in similar environments. Then, an adaptive and incremental clustering algorithm is used to group the similar runtime environment into a cluster and regard virtual machine anomaly detection in each cluster as a point anomaly detection problem. In this adaptive and incremental clustering algorithm, the number of clusters is not predefined and non-fixed. The creation of clusters is based on the criteria of maximum inter- cluster distance, minimum inner-cluster distance and the maximum dependability clustering. Experimental results show that this proposed method can adaptive to the dynamic and complexity cloud platform and detect anomalies effectively.

The rest of this paper is organized as follows. Some related works are discussed in Section 2. The definitions of cloud platform runtime environment attributes of virtual machines are presented in Section 3. The adaptive and incremental clustering algorithm for virtual machine anomaly detection under cloud platform runtime environment is proposed in Section 4. Then the experiments and results are described in Section 5. Section 6 concludes this paper.

II. RELATED WORKS

To improve the dependability of cloud platform and keep pace with the increase of scale and complexity of cloud platform, a variety of anomaly detection techniques have been proposed.

Wang *et al.* [17] proposed an Entropy based anomaly testing mechanism for the ever-increasing size and complexity of system software, applications and workload patterns of cloud platform. This mechanism used entropy as a measurement to generate entropy time series according to the distributions of performance metrics. Then online tools were used to identify anomalies in these entropy time series. Compared to the traditional anomaly detection method based on individual metric thresholds, it could effectively improve the accuracy and false alarm rate of anomaly detection system. Though the entropy based testing mechanism use hierarchically to reduce the volume of monitoring data, different levels of abstractions like hardware, software, operation system, multimedia and virtual machines are had to be collected.

In order to achieve a lightweight detection method, Wang *et al.* [18] proposed a statistical technique for online anomaly detection based on Tukey method and the multinomial goodness-of-fit test based on the Relative Entropy statistic to adapt to the needs of multiple time dimensions, entire past, recent past and context based on hour of day and day of week.

Pannu *et al.* [19] proposed a self-evolving anomaly detection framework. The health-related runtime performance data of system was monitored continuously, and a recursively statistical learning technology was exploited to calculate the abnormality score value of a new cloud performance data record. Then cloud operators pinpointed the anomalies according to the abnormality score to evolve the detection method.

For the variety failure types in cloud computing infrastructures, Guan and Fu [11] presented an adaptive anomaly identification mechanism to explore the most relevant principal components factor of different failure types. PCA method was used to find the most relevant principal components for each type of possible failures. Adaptive Kalman filters then used to do anomaly detection based on the performance metric data record.

According to the huge number of machines of cloud platform, Bhaduri *et al.* [20] proposed a distributed and automated anomaly detection framework, which runs on top of the Ganglia system on each computing machine on the cloud locally to rank the machines in order based on their anomalies or fault scores. The ranked list would be reported to the master node in the cloud platform and the system operators would take corrective actions to defend from anomalies.

Smith *et al.* [21] proposed an autonomic anomaly detection mechanism to detect anomalies from the voluminous amount of noisy and high dimensional data. The health-related system runtime performance data was collected by hardware sensors, system calls, application profiles and some third-party monitoring tools. Taking account of the data volume and diversity, data dependency, anomaly characteristics and system dynamic, Bayesian network based method and PCA method were combined to automatically analyze of the collected data record and detect the anomalies.

Fu [22] presented a performance metric selection based autonomic anomaly detection mechanism to deal with the redundancy and high dimensionality of the runtime performance metrics. Mutual information was exploited to quantify the relevance and redundancy among the performance metrics. Meanwhile, PCA method was used to further reduce the metric dimension to tackle with the high dimensionality performance metrics. Then a semi-supervised decision tree was introduced to identifies anomalies.

Aiming at the failure problem of cloud platform, Tan *et al.* [23] proposed an online predictive performance anomaly prevention system to automatic performance anomaly prevention. It integrated online anomaly prediction and virtualization-based prevention to automatically prevent performance anomalies. All of the virtual machines system performance metrics were continuously tracked by the virtual machine monitoring modules. Two-dependent Markov Chain model was incorporated with the Tree-Augmented Naïve Bayesian Network to execute anomaly detection and classification. After pinpointing the faulty virtual machines, elastic virtual machine resource scaling and live virtual migration was triggered to prevent from the performance anomalies.

In view of massive amount of the monitoring data, Gupta *et al.* [24] proposed a context-aware anomaly detection method, taking the system logs and time series data into consideration. The data instance was denoted by the combination of system logs and time series data. Then the modified K-means method was used to obtain metric patterns, and an anomaly score of an instance was obtained.

Guiping and Jiawei [25] presented an environment-aware anomaly detection framework under cloud platform for virtual machines, where the training problems of imbalance training sample sets,

multiple anomaly categories and increasing number of training samples were discussed and several SVM based anomaly detection algorithms were proposed. Liu *et al.* [26] proposed a dynamic and adaptive anomaly detection method base on self- organizing maps, where SOM network method was used to model the set of virtual machines with some similar running properties to reduce the modeling time and the training cost.

III. THE DEFINITION OF CLOUD PLATFORM RUNTIME ENVIRONMENT ATTRIBUTES

In the cloud platform, each client virtual machine with different customer application systems is scheduled by the virtual machine monitor to coordinate operation on different physical servers. The performance of each client virtual machine is affected by the status factor of physical servers and customer application systems, that is the virtual machine runtime performance metrics are closely related to its runtime environment in the cloud platform. Once the resource configurations of physical server with virtual machines running are different, the resource configurations of virtual machines will appear diversity, leading to a differ in virtual machine runtime performance metrics. Therefore, for the sake of effectively anomaly detection for virtual machines in the cloud platform, this paper proposes an adaptive and incremental clustering anomaly detection algorithm for virtual machines under the cloud platform runtime environment. A global view of the cloud platform runtime environment factors -- the resource configuration of virtual machine, the system workload of virtual machine and the resource configuration of physical server where the virtual machines runs –is given.

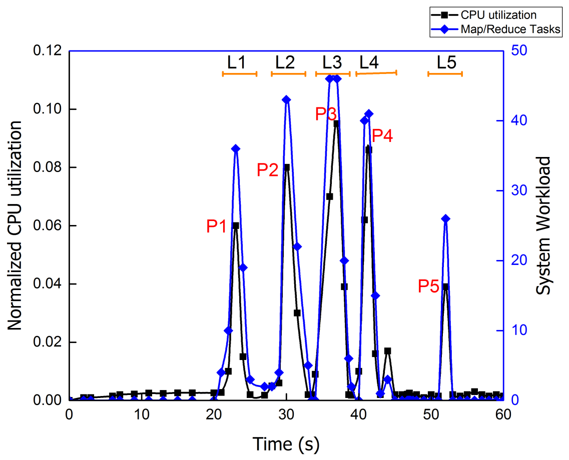


FIGURE 2. **The effect of workload on the runtime performance metrics (CPU utilization).**

Figure. 2 shows the relationships between one of the virtual machine runtime performance metrics (CPU utilization) and the system workload when executes map/reduce tasks. L1, L2, L3, L4 and L5 show the system workload of virtual machine at different time. P1, P2, P3, P4 and P5 indicate the CPU utilization at the corresponding time point. It can be seen from the Figure that under the different workloads, the virtual machine runtime performance metric index (CPU utilization) appears different. If an irregular inflection of virtual machine runtime performance metrics is identified an anomaly, then it will be considered that this virtual machine is thought to have experienced 5 anomalies within the 60 seconds. But in actually, combining with the system workload of virtual machine, the above irregular changes are normal.

Therefore, to avoid the interference of cloud platform runtime environment to anomaly detection from misjudgment, this paper takes the cloud platform runtime environment factor - the resource configuration of virtual machine, the system workload of virtual machine and the resource configuration of physical server where the virtual machines runs- into consideration, and gives the definition of each factor as follows.

Definition 1: Resource Configuration Attribute set of virtual machine (RC), which combines the CPU processing capacity, memory resource and storage source. The attribute set can be formalized as a vector, where rc_i represents one metric attribute of the vector.

$$\mathrm{RC} = [rc_1 rc_2 rc_3 \ldots rc_k]^T \tag{1}$$

Definition 2: System Workload Attribute set of virtual machine (SW), where transaction throughput and network throughput of virtual machine are considered. The attribute set can be formalized as a vector, where sw_i represents one metric attribute of the vector.

$$\mathrm{SW} = [sw_1 \quad sw_2 \quad sw_3 \quad \ldots \quad sw_h]^T \tag{2}$$

Definition 3: Server Resource Configuration Attribute set with the virtual machine running (SC). It includes the CPU frequency, memory resource, storage resource and cash size of server. The formalized vector shown as below, where sc_i represents one metric attribute of the vector.

$$\mathrm{SC} = [sc_1 \quad sc_2 \quad sc_3 \quad \ldots \quad sc_l]^T \tag{3}$$

Definition 4: The Cloud Platform Runtime Environment vector (RE), which consists of the resource configuration attribute vector RC, system workload attribute vector SW and the server resource configuration attribute vector SC together to model the cloud platform runtime environment where the virtual machine runs. The vector is shown as follows:

$$\mathrm{RE} = [\mathrm{RC}, \mathrm{SW}, \mathrm{SC}]^T \tag{4}$$

According to the vector definition of cloud platform runtime environment attribute, it can be found that there exist some relationships between each variable in the vector. For instance, the CPU processing capacity of virtual machine may have influence on the transaction throughput and the server CPU frequency may affect the CPU procession capacity of the virtual machine, too. Caring about this association, Mahalanobis Distance method are used to calculate the similarity between the runtime environments. The similarity calculation method is shown as below.

$$d_{ij} = \sqrt{(re_i - \mu_j)A^{-1}(re_i - \mu_j)^T} \tag{5}$$

Where μ_j is the center of cluster C_j, $\mu_j = \frac{\sum_{k=1}^{n} re_k}{n}$, n denotes the number of runtime environment data in cluster C_j. A is the covariance matrix of the runtime environment vectors of all the virtual machines in cluster C_j, the formula is as follows.

$$\mathrm{A} = \begin{bmatrix} cov\,(a_1, a_1) & \cdots & cov\,(a_1, a_m) \\ \vdots & \ddots & \vdots \\ cov\,(a_m, a_1) & \cdots & cov\,(a_m, a_m) \end{bmatrix} \tag{6}$$

Where a_i is the ith data vector of the runtime environment vector, and the covariance calculation formula between two attribute a_i and a_j is shown below.

$$\operatorname{cov}\left(a_i, a_j\right) = \frac{1}{n}\sum\nolimits_{k=1}^{n} (a_{ik} - \bar{a}_i)(a_{jk} - \bar{a}_j) \tag{7}$$

where $\bar{a}_i$ is the average value of ith attribute, $\bar{a}_i = \frac{1}{n}\sum_{k=1}^{n} a_{ik}$.

IV. THE PROPOSED ADAPTIVE AND INCREMENTAL CLUSTERING ANOMALY DETECTION ALGORITHM

Clustering is a common unsupervised learning method, which is based on the similarity and groups the data into a cluster set according to a certain similarity measure without any prior knowledge. Due to the characteristics, clustering methods such as K-means [27], phased k-means [28], k-medoids [29], [30], k-nearest neighbors [31] and Kernel K-means [32] based method have been widely used in anomaly detection in the cloud platform.

Nevertheless, either the k-means based anomaly detection method or the k-nearest neighbors based anomaly detection method, it has to predefined the number of cluster k. Besides, a randomly initialization of each cluster center is further needed for cluster-based anomaly detection method. But cloud platform is a continues and dynamic operation environment, a long time durative monitoring is required to ensure the stability and dependability of the cloud platform.

As described in Section 3, the cloud platform runtime environment factor has a major influence on virtual machine anomaly detection. During the operating time, there may be a new runtime environment circumstance. A stable and predefined cluster number k might not adapt to the dynamic cloud platform.

Accordingly, to improve the detection accuracy, detection speed and the adaptability of anomaly detection for virtual machine in the cloud platform, the proposed adaptive and incremental clustering anomaly detection algorithm do not predefine and fix the cluster number. Each input data vector will be incrementally assigned to the most similarity cluster over time, and the number of clusters will be monotonically increased. In order to make the cluster results reliable, three criteria have to be satisfied: maximal inter-cluster distance, minimal inner-cluster distance and maximal dependability clustering rules.

Minimal inner-cluster distance means that the data points in the same cluster has the maximal similarities. Using the mean value of all data points to the cluster center, it can be specified as:

$$\min \frac{1}{|C_i|} \sum\nolimits_{v \in C_i} d(v, \mu_i) \tag{8}$$

Where $|C_i|$ is the number of data points in cluster C_i, v is the data point belongs to the cluster C_i and μ_i is the center of the cluster C_i.

Maximum inter-cluster distance means that the data points in two different clusters are far from each other. A data point in one cluster will not possible belong to another cluster at all. It can be specified as:

$$\max \frac{1}{|C_i|\,|C_j|} \sum\nolimits_{\substack{v \in C_i \\ u \in C_j}} d(v, u) \tag{9}$$

Where $|C_i|$, $|C_j|$ represent the number of data points in cluster C_i and C_j respectively.

Combine with these two criteria, the maximal dependability clustering creation rules of the adaptive and incremental clustering algorithm are as follows. First, the similarity of each data point in the input vector is calculated by the Mahalanobis Distance. Then, the maximal similarity threshold value $SIM_{threshold}$ is used to decide whether a new cluster is created or not. If the similarity between a new data point v and one cluster C_k is larger than $SIM_{threshold}$ then assigns the cluster index k to data point v. Otherwise, the cluster number is increased with 1. Once a data point v get the cluster index k, the update process of cluster C_k needs to be executed. Whereas, considering the flexibility of cloud platform and the associations of each metric attribute, although the data point v is similar to cluster center μ_k, it may be far away from the rest data point of cluster C_k. A simple update process of cluster center μ_k with the average value of all values between data point v and cluster C_k will result in a lateral shift of the cluster C_k. Here, a comparison between the similarity of data point v with cluster center μ_k and the similarity of data point v with all the other data point in cluster C_k is performed. The maximal dependable distance update threshold value $DEPEND_{threshold}$. If the difference between them is larger than the $DEPEND_{threshold}$, then the cluster number is increased with 1. Otherwise, the cluster center μ_k is updated by the average value of all values between data point v and cluster C_k.

Algorithm 1 presents the pseudocode of the proposed clustering algorithm for incrementally and adaptively creating and updating clusters.

Algorithm 1 Incrementally and Adaptively Creating and Updating Clusters

```
Procedure BuildCluster(inputVector)
For each $v$ in inputVector do
    numOfCluster ← 0
clusterIndex ← GetTheClusterIndex($v$)
if clusterIndex == −1 then
    CreateNewCluster($v$)
    numOfCluster ← numOfCluster+1
else
      numOfCluster ← UpdateCluster($v$, clusterIndex, numOfCluster)
Procedure GetTheClusterIndex($v$)
clusterIndex ← −1
maxSimilarity ← MAX_NEG
for each cluster k do
    distance ← MahalanobisDis($v$, $\mu_k$)
    similarity ← $\frac{1}{1+distance}$
    if similarity > $SIM_{threshold}$ then
        if similarity > maxSimilarity then
maxSimilarity ← similarity
clusterIndex ← $k$
return clusterIndex
Procedure UpdateCluster($v$, clusterIndex, numOfCluster)
$C_i$ ← GetTheSimiarityCluster(clusterIndex)
$\mu_i$ ← GetTheClusterCenterByIndex($C_i$)
dependability ← d ($v$, $\mu_i$) − $\frac{1}{|C_i|} \sum_{u \in C_i} d(v, u)$
if dependability > $DEPEND_{threshold}$ then
    CreateNewCluster($v$)
    numOfCluster ← numOfCluster+1
else
    UpdateClusterCenter($v$, $C_i$)
return numOfCluster
```

V. PERFORMANCE EVALUATION

A. EXPERIMENTAL SETUP

In this paper, the open source cloud platform OpenStack [33], [34] was used to build the experimental cloud platform. This cloud platform consists of 15 servers, which are connected by gigabit Ethernet. All the servers are installed the operation system CentOS6.5. The hypervisor Xen3.2 [35], [36] is installed on the servers for running virtual machines. The cloud managements component OpenStack are installed on the server for running the cloud management program. All the servers are equipped with two to four Intel Xeon Opteron cores and 4 to 8GB of RAM. Each server hosts up to eight virtual machines (VMs). the runtime performance metrics set of VMs in this platform is collected by the third-party tools libxenstat and libvirt [37], [38]. And during the runtime, three types of system failures are simulated by fault injection tools, that is CPU Hog, network Hog and memory leak [39]–[41].

B. EXPERIMENTAL PROGRAM AND RESULTS

1) FIRST SET OF EXPERIMENTS

The performance of the adaptive and incremental clustering anomaly detection algorithm for virtual machines under the cloud platform runtime environment (AI-Cluster for short) was evaluated on the stable running environment. In order to evaluate the effectiveness of the proposed algorithm, two typical unsupervised anomaly detection techniques are used for comparison in the experiments: (1) k-nearest neighbor based anomaly detection technique (k-NN for short); (2) cluster-based (k-means) anomaly detection technique (k-Means for short).

Training Stage: First, 100 virtual machines were deployed on the server. To simulate a stable runtime environment to exclude the impact of the runtime environment factors, the load testing tool LoadRunner [42]–[44] was used to apply the same workload to each virtual machine. Then, several virtual machines were chosen from these 100 virtual machines in random. At the same time, one of the three fault types was selected to injected into these virtual machines. A total of 1000 virtual machine runtime performance metrics were collected from the 100 virtual machines during 10 rounds (one second per round) collection process as the training samples to train detection model.

Detection Stage: In order to simulate an anomaly behavior of the detection objects (virtual machines), three faults are randomly injected into the 100 virtual machines. The anomalies were then detected by the trained model as well as the detection results were recorded.

Several sets of experimental results with different fault types injection methods were obtained. It should be noted that there is no need to do the train process of the proposed AI-Cluster algorithm and the k-NN algorithm, which start the anomaly detection stage directly in the experiment. While for the k- means based anomaly detection method the training process is required. To ensure comparability, the fault types and the testing data set in the detection stage is the same. Experimental results of anomaly detections in the presence of different fault types respectively are shown in Table 1.

The results in Table 1 indicate the proposed AI-Cluster algorithm shows a great robustness to identify anomalies in the presence of different fault types and outperforms the other two unsupervised anomaly detection algorithms. In addition, it can be seen from the Table 1 that compared to the CPU Hog and Network Hog failures, the correctly detection rate of Memory Leak is relatively low. Due to the fault caused by the memory leak does not lead to an anomaly behavior of the failed object immediately in contrast to the faults caused by CPU Hog and network Hog, it will result in a high false alarm rate and poor detection accuracy. To illustrate the detection performance, the ROC curves of the anomaly detection algorithms is used to show the trade-off between the true positive rate and false positive rate of the algorithms in Figure 3.

TABLE 1. Performance comparison of algorithms in the presence of different faults.

Fault types	Algorithm	Conversion Correctly Detection Rate(%)
CPU Hog	k-NN	56.80
	k-Means	68.02
	AI-Cluster	89.93
Memory Leak	k-NN	41.49
	k-Means	44.85
	AI-Cluster	83.33
Network Hog	k-NN	85.00
	k-Means	81.11
	AI-Cluster	95.55

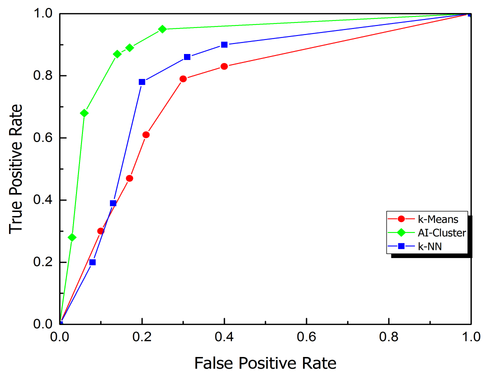


FIGURE 3. The ROC curves of detection methods under the same runtime environment.

In Figure 3, the true positive rate is the proportion of the anomalies that are correctly identified, and the false positive rate is the proportion of the normal instances that are identified as abnormities. From it, the proposed AI-Cluster method shows the most promising and competitive accurate rates compared with the performance of k-NN method and k-Means method.

2) SECOND SET OF EXPERIMENTS

The objective of this set of experiments was to evaluate the performance of the proposed adaptive and incremental clustering anomaly detection algorithm on the dynamic changing runtime environment. Here, the aforementioned typical unsupervised anomaly detection techniques were used too.

100 virtual machines were still deployed in the cloud platform to simulate the detection objects. For the sake of convenience, the changing of runtime environment that the virtual machines running was achieved by the LoadRunner. The change rate of the runtime environment was obtained by modifying a number of virtual machines system workload. To estimate the effect of the environment factor, the training data set of k-means method is the same as the first set of experiments. The change rate and the detection results were record.

The experimental results are shown in Table 2, Table 3 and Figure 4.

Table 2 tabulates the detection accuracy rate of detection methods under the changing environment. The results show that there is no obvious change in accuracy using the proposed AI-Cluster detection method in the presence of different runtime environment, which means the proposed AI-Cluster detection method could adaptive to the dynamic cloud platform runtime environment.

TABLE 2. Comparison of the accuracy of anomaly detection methods under the changing environment.

Change rate of Runtime Environment (%)	Accuracy Rate(%)		
	k-NN	k-Means	AI-Cluster
10	≈97.7	≈95.7	≈98.7
20	≈86.8	≈87.8	≈93.8
30	≈83.2	≈81.2	≈89.2
40	≈79.8	≈67.8	≈88.8
50	≈83.6	≈62.6	≈87.6

TABLE 3. Comparison of the average detection latency of anomaly detection methods under the changing environment.

Change rate of Runtime Environment (%)	Average Detection Latency(ms)		
	k-NN	k-Means	AI-Cluster
10	20	10	21
20	25	19	20
30	21	25	22
40	29	37	21
50	31	50	22

Table 3 tabulates the detection latency of the detection methods with the runtime environment changing. It can be seen from Table 3 that when the environment change rate is lower, the AI-Cluster method and k-NN method have a higher latency than k-means. The main reason is that AI-Cluster method and k-NN method are data-driven anomaly detection method, there is no need of training process for model training, while the k- means based model has. As the change rate of runtime environment increasing, AI-Cluster detection method shows the smoothest detection latency followed by the k-NN detection method with the k-means based detection method in the last.

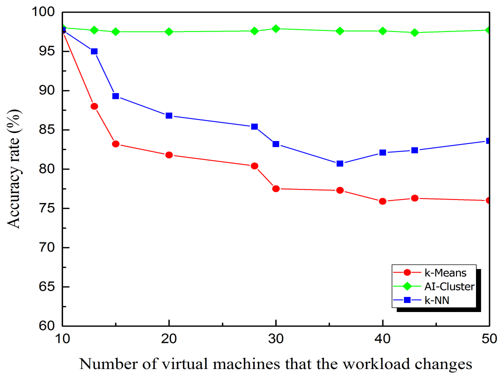


FIGURE 4. The comparison of accuracy rates of detection methods in concept of workload changing.

Figure. 4 further illustrates the comparison of the accuracy rates of the anomaly detection methods under the changing runtime environment. In the figure, the horizontal axis represents the numbers of the virtual machines that the workload changes with the vertical axis indicating the detection accuracy rates. In order to compare the performance in real concept of workload changing, the same fault types were injected at the same change rate of the runtime environment. The evaluation results shown in these figures indicate that the proposed AI-Cluster algorithm performs significantly well in the anomaly detection for virtual machines under the high dynamic changing environment.

Overall, the above experimental results show that the proposed adaptive and incremental clustering algorithm for virtual machines under cloud platform runtime environment (AI-Cluster for short) has great flexibility and adaptability in the elastic cloud platform environment in comparison with the k-nearest neighbors based anomaly detection technique and the cluster-based (k-means) anomaly detection technique. The incremental mechanism of the proposed AI-Cluster method and the rules it used to create clusters make it easy to aware of the changing runtime environment and adapt itself. The experimental results prove that it outperforms the commonly used anomaly detection methods and improves detection accuracy rates greatly in the dynamic cloud platform.

VI. CONCLUSION

In this paper, an adaptive and incremental clustering anomaly detection algorithm for virtual machines under cloud platform runtime environment has been presented. Different from the other common anomaly detection techniques in cloud platform, this method takes the elastic cloud platform runtime environment into consideration. Before executing detection, the runtime environment attributes set vector as well as the similarity calculation method between each vector instance is defined. Then the ideas of this proposed anomaly detection method is presented followed by the corresponding pseudo code. Finally, two sets of experiments are performed to evaluate the performance of the proposed anomaly detection method on the real cloud platform. The experimental results prove that this adaptive and incremental clustering anomaly detection algorithm for virtual machines under the cloud platform runtime environment can efficiently detect the arrival of anomaly instance and can also greatly improve the detection accuracy rate under different runtime environments.

In the future, we aim to implement the qualitative and quantitative description models of the cloud platform runtime environment. A round model will help to improve the detection accuracy and dependability of the detection method. In addition, to further extend the capabilities of the current adaptive and incremental clustering anomaly detection method, the fusion and analysis of heterogeneous data from a variety of sources with be a trend.

REFERENCES

[1] G. Kousiouris, A. Menychtas, D. Kyriazis, S. Gogouvitis, and T. Varvarigou, "Dynamic, behavioral-based estimation of resource provisioning based on high-level application terms in cloud platforms," *Future Gener. Comput. Syst.*, vol. 32, no. 2, pp. 27–40, 2014.

[2] J. A. Wickboldt, R. P. Esteves, M. B. de Carvalho, and L. Z. Granville, "Resource management in IaaS cloud platforms made flexible through programmability," *Comput. Netw.*, vol. 68, no. 5, pp. 54–70, 2014.

[3] Z. Zhuang, H. Ramachandra, and B. Sridharan, "Guarding fast data delivery in cloud: An effective approach to isolating performance bottleneck during slow data delivery," in *Cloud Computing and Big Data* (Lecture Notes in Computer Science), vol. 9106. Cham, Switzerland: Springer, 2015.

[4] S. Kikuchi and K. Hiraishi, "Improving reliability in management of cloud computing infrastructure by formal methods," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–7.

[5] F. Langner and A. Andrzejak, "Detecting software aging in a cloud computing framework by comparing development versions," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, May 2013, pp. 896–899.

[6] R. Kumar, S. Vijayakumar, and S. A. Ahamed, "A pragmatic approach to predict hardware failures in storage systems using MPP database and big data technologies," in *Proc. IEEE Adv. Comput. Conf.*, Feb. 2014, pp. 779–788.

[7] S. N. Brohi, M. A. Bamiah, M. N. Brohi, and R. Kamran, "Identifying and analyzing security threats to virtualized cloud computing infrastructures," in *Proc. Int. Conf. Cloud Comput. Technol., Appl. Manage.*, Dec. 2012, pp. 151–155.

[8] B. Dragovic *et al.*, "Xen and the art of virtualization," *ACM SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, 2009.

[9] C.-T. Yang, J.-C. Liu, C.-H. Hsu, and W.-L. Chou, "On improvement of cloud virtual machine availability with virtualization fault tolerance mechanism," *J. Supercomput.*, vol. 69, no. 3, pp. 1103–1122, 2014.

[10] Y. Jin, Y. Wen, Q. Chen, and Z. Zhu, "An empirical investigation of the impact of server virtualization on energy efficiency for green data center," *Comput. J.*, vol. 56, no. 8, pp. 977–990, 2013.

[11] Q. Guan and S. Fu, "Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures," in *Proc. IEEE Int. Symp. Reliable Distrib. Syst.*, Sep. 2013, pp. 205–214.

[12] G. Wang, S. Chen, and J. Liu, "Anomaly-based intrusion detection using multiclass-SVM with parameters optimized by PSO," *Int. J. Secur. Appl.*, vol. 9, no. 6, pp. 227–242, 2015.

[13] 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 *Proc. IEEE Res. Develop.*, Nov. 2009, pp. 89–92.

[14] L. Nie, D. Jiang, and Z. Lv, "Modeling network traffic for traffic matrix estimation and anomaly detection based on Bayesian network in cloud computing networks," *Ann. Telecommun.*, vol. 72, nos. 5–6, pp. 297–305, 2016.

[15] A. A. Sebyala, T. Olukemi, L. Sacks, and D. L. Sacks, "Active platform security through intrusion detection using naive Bayesian network for anomaly detection," in *Proc. London Commun. Symp.*, 2002, pp. 1–5.

[16] W. Sha, Y. Zhu, M. Chen, and T. Huang, "Statistical learning for anomaly detection in cloud server systems: A multi-order Markov chain framework," *IEEE Trans. Cloud Comput.*, vol. 6, no. 2, pp. 401–413, Apr. 2018.

[17] C. Wang, V. Talwar, K. Schwan, and P. Ranganathan, "Online detection of utility cloud anomalies using metric distributions," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2010, pp. 96–103.

[18] C. Wang, K. Viswanathan, C. Lakshminarayan, V. Talwar, W. Satterfield, and K. Schwan, "Statistical techniques for online anomaly detection in data centers," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, May 2011, pp. 385–392.

[19] H. S. Pannu, J. Liu, and S. A. Fu, "A self-evolving anomaly detection framework for developing highly dependable utility clouds," in *Proc. Global Commun. Conf. (GLOBECOM)*, Dec. 2012, pp. 1605–1610.

[20] K. Bhaduri, K. Das, and B. L. Matthews, "Detecting abnormal machine characteristics in cloud infrastructures," in *Proc. Data Mining Workshops (DBLP)*, Dec. 2011, pp. 137–144.

[21] D. Smith, Q. Guan, and S. Fu, "An anomaly detection framework for autonomic management of compute cloud systems," in *Proc. Workshop IEEE Int. Comput. Softw. Appl. Conf. (COMPSAC)*, Seoul, South Korea, Jul. 2010, pp. 376–381.

[22] S. Fu, "Performance metric selection for autonomic anomaly detection on cloud computing systems," in *Proc. Global Commun. Conf. (GLOBECOM)*, Houston, TX, USA, Dec. 2011, pp. 1–5.

[23] Y. Tan, H. Nguyen, Z. Shen, X. Gu, C. Venkatramani, and D. Rajan, "PREPARE: Predictive performance anomaly prevention for virtualized cloud systems," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, Jun. 2012, pp. 285–294.

[24] M. Gupta, A. B. Sharma, H. Chen, and G. Jiang, "Context-aware time series anomaly detection for complex systems," in *Proc. SDM Workshop Data Mining Service Maintenance*, 2013, pp. 14–22.

[25] G. Wang and J. Wang, "An anomaly detection framework for detecting anomalous virtual machines under cloud computing environment," *Int. J. Secur. Appl.*, vol. 10, no. 1, pp. 75–86, 2016.

[26] J. Liu, S. Chen, Z. Zhou, and T. Wu, "An anomaly detection algorithm of cloud platform based on self-organizing maps," *Math. Problems Eng.*, vol. 2016, no. 1, 2016, Art. no. 3570305.

[27] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proc. 18th ACM-SIAM Symp. Discrete Algorithms*, New Orleans, LA, USA, Jan. 2015, pp. 1027–1035.

[28] U. Rebbapragada, P. Protopapas, C. E. Brodley, and C. Alcock, "Finding anomalous periodic time series," *Mach. Learn.*, vol. 74, no. 3, pp. 281–313, 2009.

[29] S. Budalakoti, A. N. Srivastava, R. Akella, and E. Turkov, "Anomaly detection in large sets of high-dimensional symbol sequences," NASA, Moffett Field, CA, USA, Tech. Rep. NASA/TM-2006-214553, 2006.

[30] S. Budalakoti, A. N. Srivastava, and M. E. Otey, "Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 39, no. 1, pp. 101–113, Jan. 2009.

[31] V. Hautamaki, I. Karkkainen, and P. Franti "Outlier detection using k-nearest neighbour graph," in *Proc. Int. Conf. Pattern Recognit.*, vol. 3, Aug. 2004, pp. 430–433.

[32] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: Spectral clustering and normalized cuts," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 551–556.

[33] D. Milojičić, I. M. Llorente, and R. S. A. Montero, "OpenNebula: A cloud management tool," *IEEE Internet Comput.*, vol. 15, no. 2, pp. 11–14, Mar. 2011.

[34] J. M. A. Calero and J. G. Aguado, "MonPaaS: An adaptive monitoring platformas a service for cloud computing infrastructures and services," *IEEE Trans. Services Comput.*, vol. 8, no. 1, pp. 65–89, Jan. 2014.

[35] H. Jin, H. Qin, S. Wu, and X. Guo, "CCAP: A cache contention-aware virtual machine placement approach for HPC cloud," *Int. J. Parallel Program.*, 2015, vol. 43, no. 3, pp. 403–420.

[36] B. Egger, E. Gustafsson, C. Jo, and J. Son "Efficiently restoring virtual machines," *Int. J. Parallel Program.*, vol. 43, no. 3, pp. 421–439, 2015.

[37] Y. Cho, J. Choi, and J. Choi, "Towards an integrated management system based on abstraction of heterogeneous virtual resources," *Cluster Comput.*, vol. 17, no. 4, pp. 1215–1223, 2014.

[38] J. Li, Y. Jia, and L. Liu, "CyberLiveApp: A secure sharing and migration approach for live virtual desktop applications in a cloud environment," *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 330–340, 2013.

[39] Y.-J. Chin and T. Berger, "A software-only videocodec using pixel-wise conditional differential replenishment and perceptual enhancements," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 3, pp. 438–450, Apr. 1999.

[40] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 743–761, Apr. 2012.

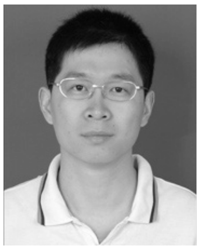
[41] Z. Xu, J. Zhang, and Z. Xu "Melton: A practical and precise memory leak detection tool for C programs," *Frontiers Comput. Sci.*, vol. 9, no. 1, pp. 34–54, 2015.

[42] S. Nachiyappan and S. Justus, "Cloud testing tools and its challenges: A comparative study," *Proc. Comput. Sci.*, vol. 50, pp. 482–489, Jan. 2015.

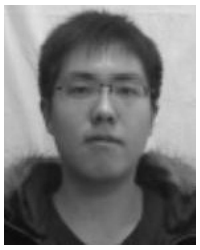
[43] X. J. Dai and N. Zhang, "Performance testing and optimization of data analysis platform based on LoadRunner," *Comput. Technol. Develop.*, vol. 23, no. 3, pp. 202–206, 2013.

[44] M. Kaushik, "Research of load testing and result based on loadrunner," *Comput. Sci.*, vol. 6, no. 4, pp. 301–310, 2014.

HANCUI ZHANG received the Ph.D. degree from the School of Big Data and Software Engineering, Chongqing University, China, in 2018. Since 2018, she has been a Teacher with the School of Information Science and Technology, Zhejiang Sci-Tech University, China. Her current interests include cloud computing, large-scale data mining, flash memory, and fault detection.



JUN LIU received the Ph.D. degree from Chongqing University, China, in 2017. He is currently a Teacher with the College of Software Engineering, Chongqing University of Posts and Telecommunications. His current interests include large-scale data mining, flash memory, networking security, and fault detection.



TIANSHU WU received the B.S. degree from the Chongqing University of Posts and Telecommunications, China, in 2011. He is currently pursuing the Ph.D. degree with the College of Computer Science, Chongqing University. His current interests include cloud computing, big data, and fault detection.