International Journal of Security and Its Applications
Vol. 10, No. 1 (2016), pp.75-86
http://dx.doi.org/10.14257/ijsia.2016.10.1.08

An Anomaly Detection Framework for Detecting Anomalous Virtual Machines under Cloud Computing Environment

GuiPing Wang* and JiaWei Wang

College of Information Science and Engineering, Chongqing Jiaotong University, Chongqing, China
w_guiping@163.com

Abstract

A variety of faults may cause performance degradation or even downtime of virtual machines (VMs) under Cloud environment, thus lowering the dependability of Cloud platform. Detecting anomalous VMs before real failures occur is an important means to improve the dependability of Cloud platform. Since the performance or state of VMs may be affected by the environmental factors, this article proposes an environment-aware anomaly detection framework (termed EaAD) for VMs under Cloud environment. EaAD partitions all the VMs in Cloud platform into several monitoring domains based on similarity in running environment, which makes the VMs in a same monitoring domain have similar running environment. In each domain, the equipped anomaly detection algorithm detects anomalous VMs based on their performance metrics. In addition, anomaly detection in a certain monitoring domain faces such challenges as multiple anomaly categories, imbalanced training sample sets, increasing number of training samples. To cope with these challenges, several support vector machine (SVM) based anomaly detection algorithms are implemented and equipped in EaAD, including C-SVM, OCSVM, multi-class SVM, imbalanced SVM, online learning SVM. This article conducts experiments on EaAD to test the performance of the adopted detection algorithms and looks into future work.

Keywords: *Cloud Computing; Virtual Machine (VM); Environment-aware; Anomaly Detection; Support Vector Machine (SVM)*

1. Introduction

Along with the increasing scale and complexity of Cloud platforms, various deliberate or non-deliberate faults may cause frequent performance degradation or even downtime accidents of virtual machines (VMs), which greatly lowers the dependability of Cloud platforms. These faults usually include hardware faults ([1]), software defects ([2]), mis-configuration ([3]), and attacks ([4]).

In general, the abnormal states of VMs appear before real failures occur under Cloud environment. Further, the performance or state of a VM can be characterized by performance metrics, while the value of a performance metric is also affected by the environmental factors including resource configuration of the VM, the workload of the VM, and the resource configuration of the underlying physical host. A same VM will exhibit different performance under different running environment. The anomaly detection system should exclude the performance deviation between VMs caused by different running environment. Otherwise, it will result in a large number of false positives.

Therefore, this article proposes an environment-aware anomaly detection framework

* Corresponding Author

ISSN: 1738-9976 IJSIA

(termed EaAD) for VMs under Cloud environment. Concretely, EaAD first partitions all the VMs in Cloud platform into several monitoring domains based on similarity in running environment, so as to make VMs in a same monitoring domain have similar running environment. In each monitoring domain, the equipped anomaly detection algorithm detects anomalous VMs based on their performance metrics and reports to the operator.

In addition, anomaly detection of VMs in a certain monitoring domain under Cloud environment faces such challenges as multiple anomaly categories, imbalanced training sample sets, increasing number of training samples. To cope with these challenges, this article implements several SVM-based anomaly detection algorithms in literature and equip them in EaAD. These algorithms include C-SVM ([5][6]), OCSVM ([7]), multi-class SVM ([8][9]), imbalanced SVM ([10]), online learning SVM ([11][12]). This article also designs strategies of selecting anomaly detection algorithms for different situations.

The remainder of this article is organized as follows. Section 2 introduces related work. Section 3 gives fundamental definitions and the detection principles of EaAD. Section 4 presents EaAD in detail. Section 5 conducts experiments. Section 6 concludes this article and looks into future work.

2. Related Work

In order to improve the dependability of the Cloud computing infrastructure, Pannu *et al*. ([13]) present an adaptive failure detection (AFD) framework. AFD monitors the health states of physical servers, collects runtime performance metrics data of VMs, and sends them to AFD. AFD characterizes the behaviors of VMs, identifies possible failure states, and reports them to operators for verification. The verified detection results will be input back to AFD for adaptation. AFD constructs a hypersphere to cover the majority of sample points, while those sample points outside the hypersphere are identified as possible failures. However, a single hypersphere cannot accurately characterize VMs' complicated and diversified behaviors. In addition, AFD cannot deal with multiple categories of anomalies.

Lan *et al*. ([14]) present an automated mechanism for node-level anomaly identification in large-scale systems. Health-related performance metric data (*e.g*., CPU utilization, available memory size, I/O, network traffic) are collected across the system for anomaly identification. Node grouping dynamically divides system resources into groups and the nodes in the same group are expected to exhibit similar behaviors. Data transformation, feature extraction, and outlier detection, are applied per group to find abnormal nodes (*i.e*., anomalies). The detected anomalies are manually validated by system administrators. However, they do not clarify how to dynamically group nodes.

A typical Cloud computing system is essentially service-oriented. The response time of user requests directly reflects system performance. Mi *et al*. ([15]) present a framework - CloudDiag, for Cloud computing systems, which diagnoses performance by tracing user requests. CloudDiag periodically collects the end-to-end tracing data (especially, execution time of method invocations), and employs a customized Map-Reduce algorithm to proactively analyze the tracing data. However, the collected monitoring data in CloudDiag cannot fully reflect the state of Cloud computing systems.

Guan *et al*. ([16]) design a Cloud dependability analysis (CDA) framework. Fault injection agents inject random or specific faults into multiple layers (*e.g*., hypervisor, VMs, and applications). Health monitoring sensors which reside on each cloud server periodically record a set of health related performance metrics. The main purpose of CDA is to analyze the correlation of various performance metrics with failure occurrences in virtualization and non-virtualization environments, so as to gain insight into the impact of virtualization on the cloud dependability. Therefore, the function of CDA is not comprehensive enough. It cannot automatically detect anomalous VMs.

Wang *et al*. ([17]) propose EbAT - Entropy-based Anomaly Testing for online detection of anomalies in datacenters and utility Clouds. EbAT detects anomalies by analyzing the distribution of arbitrary metric. Entropy is used to characterize the degree of dispersal or concentration of such distribution. The adopted anomaly detection method is not comprehensive enough for Cloud environment.

Meng *et al*. ([18]) present a WIndow-based StatE monitoring (WISE) framework for state monitoring in Cloud datacenters. WISE reports alerts only if state violation is continuous within a time window, thus saving both much bandwidth and considerable CPU cycles. However, they do not clarify the relation between state violation and dependability threats (*i.e*., fault, error, anomaly, and failure).

In sum, the anomaly detection framework under Cloud environment faces such challenges as resource consumption of the detection framework itself, high-dimensionality and diversity of data, massive data, the dependency among data, complexity and diversity of anomalies, and the labels of samples. Despite the above plentiful researches in designing anomaly detection frameworks under Cloud environment, a framework for accurately detecting anomalous VMs in real time under large-scale and high-dynamic Cloud environment is still a challenging work.

3. Definitions and Detection Principles

The fundamental definitions and the detection principles adopted in EaAD are given as follows.

Definition 1 (Anomaly): An anomaly refers to the state of a detected system (or a node in the system) that deviates from the expected normal states, or deviates from the states of most of the time (or most of other nodes in the system).

Definition 2 (Anomaly Detection): Anomaly detection is a function of detecting the abnormal states of a detected system or abnormal nodes in a detected system.

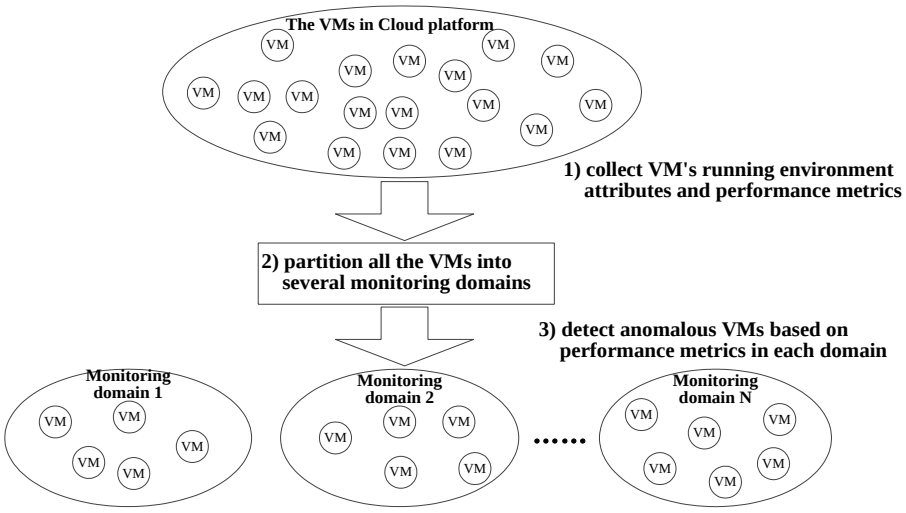


Figure 1. The Detection Principles of EaAD

In order to implement environment-aware detection and improve the detection accuracy, EaAD follows the following detection principles, as shown in Figure 1.

1) EaAD collects all the VM's running environment attributes and performance metrics at the same time.
2) EaAD partitions all the VMs in Cloud platform into several monitoring domains based on similarity in running environment by using clustering algorithms, which makes VMs in a same monitoring domain have similar running environment.
3) In each domain, the equipped anomaly detection algorithm detects anomalous VMs

based on their performance metrics.

According to the above detection principles, two important definitions are given below.

Definition 3 (VM's Running Environment Attribute Set): This attribute set includes resource configuration of a VM, the workload of the VM, and the resource configuration of the underlying physical host. The attribute set can be formalized by the following vector:

$$\boldsymbol{R} = [R_1 \ R_2 \ \ldots \ R_r]^T, \tag{1}$$

Where R_i represents an environment attribute, r is the number of attributes.

Definition 4 (VM's Performance Metric Set): This set includes a set of metrics. Each metric is an individually measurable variable, which characterizes the performance or state of a VM from a certain point of view. The metric set can be formalized by the following vector:

$$\boldsymbol{X} = [X_1 \ X_2 \ \ldots \ X_n]^T, \tag{2}$$

Where X_i represents a performance metric, n is the number of metrics.

The sample matrix (defined below) of VMs in a certain monitoring domain is the important data source of anomaly detection algorithms and other relative algorithms.

Definition 5 (Sample and Sample Matrix): Assume that an observed VM has n performance metrics, X_i, $i = 1, \ldots, n$. Each metric can be considered as a random variable. These n metrics constitute a random vector, $\boldsymbol{X}$. All the sample values of X_i ($i = 1, \ldots, n$) in a point-in-time constitute a sample of $\boldsymbol{X}$ (denoted as $\boldsymbol{x}$). Further, assume that totally l samples of all VMs in a monitoring domain are obtained in a certain time period. These l samples constitute an n-by-l original sample matrix, $\boldsymbol{X}_{n\times l}$, as shown in Eq. (1), where each column ($\boldsymbol{x}_i$) represents a sample of all metrics of a VM in a point-in-time.

$$\boldsymbol{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}, \ \boldsymbol{X}_{n\times l} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1l} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2l} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \cdots & x_{nl} \\ \uparrow & \uparrow & \uparrow & & \uparrow \\ \boldsymbol{x}_1 & \boldsymbol{x}_2 & \boldsymbol{x}_3 & \cdots & \boldsymbol{x}_l \end{bmatrix}. \tag{3}$$

For the problem of detecting anomalous VMs in a certain monitoring domain, this article gives the following formal definition.

Problem Definition (Anomaly Detection of VMs): Let T be a training sample set representing the samples of all the VMs in a monitoring domain for a certain time period,

$$T = \{ (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_l, y_l) \} \in (R^n \times \Psi)^l, \tag{4}$$

Where $\boldsymbol{x}_i \in R^n$ is the input vector (or instance), y_i is the output (or the label of $\boldsymbol{x}_i$), ($\boldsymbol{x}_i$, y_i) is called a sample point, l is the number of samples.

1) Binary classification: the task is to determine whether the state of a VM represented by a sample is normal or abnormal, then

$$y_i \in \Psi = \{1, -1\}, \ i = 1, 2, \ldots, l. \tag{5}$$

When $y_i = +1$, $\boldsymbol{x}_i$ is called a positive sample; while when $y_i = -1$, $\boldsymbol{x}_i$ is called a negative sample. The goal is to find a real function $g(\boldsymbol{x})$ in R^n,

$y = f(\boldsymbol{x}) = \text{sgn}(g(\boldsymbol{x}))$, (6)

Such that $f(\boldsymbol{x})$ derives the value of y for any sample $\boldsymbol{x}$, where sgn() is the sign function.

2) Multiclass classification: the task is to not only determine whether the state of a VM is normal or abnormal, but also determine the type of anomaly, then

$y_i \in \Psi = \{ 1, 1, ..., c \}, i = 1, 2, \ldots, l,$ (7)

c is the number of states including the normal state. The goal is to find a decision function $f(\boldsymbol{x})$ in R^n,

$y = f(\boldsymbol{x}) : R^n \rightarrow \Psi,$ (8)

Such that the class label y of any sample $\boldsymbol{x}$ can be predicted by $y = f(\boldsymbol{x})$.

4. The Proposed Anomaly Detection Framework - EaAD

This section presents EaAD in detail, including the description of equipped anomaly detection algorithms in EaAD and strategies of selecting algorithms.

4.1. EaAD

Figure 2 illustrates EaAD, which is composed of several modules, including *Partition & Deployment*, *Collection & Transmission*, *Data Processing*, *Environment-aware Detection*, and *Candidate VMs Detection*. Each module contains several function modules.

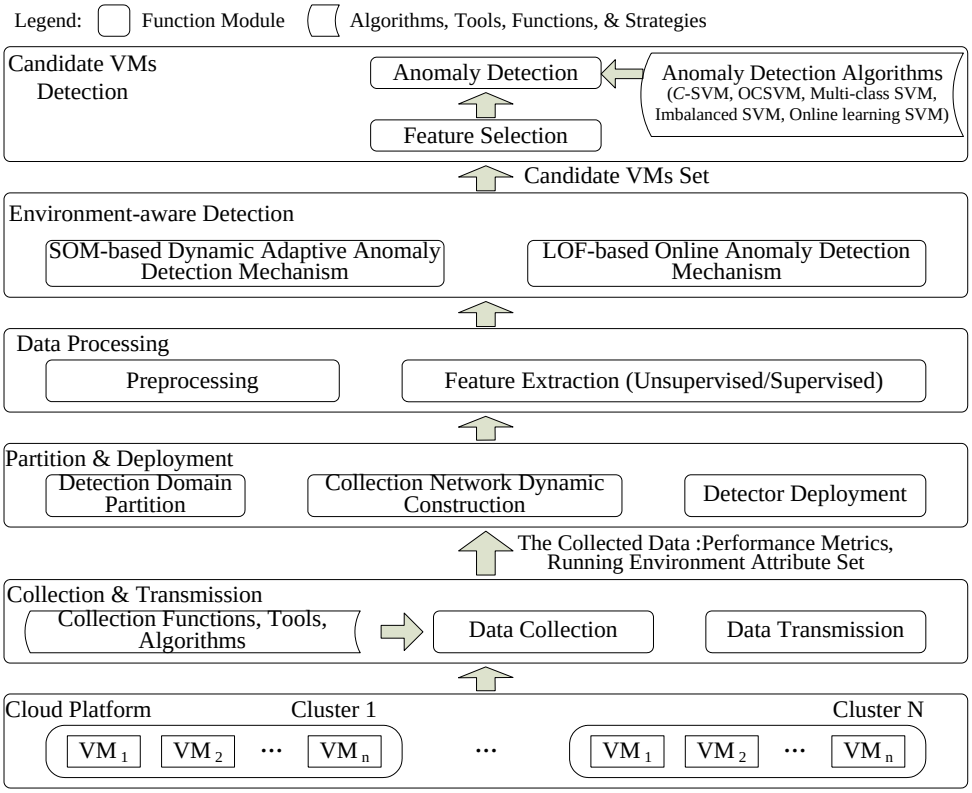


Figure 2. The Proposed Framework - EaAD

The function of each module is detailed as follows.

Collection & Transmission is responsible for collecting the performance metric data and running environment attribute data of all VMs and transmitting to the upper module.

Partition & Deployment is responsible for partitioning all the VMs into several

monitoring domains according to VMs' running environment attribute set, constructing dynamic collection network, and deploying detectors for each monitoring domain.

Data Processing is responsible for indispensable processing including feature extraction on collected data before anomaly detection.

Environment-aware Detection is responsible for preliminarily detecting anomalous VMs and submitting the set of candidate VMs to the upper module for further detection. This module adopts SOM (Self Organizing Map) based or LOF (Local Outlier Factor) based anomaly detection mechanism (optionally).

Candidate VMs Detection is responsible for further detecting anomalous VMs using more precise and powerful detection algorithms (*i.e.*, SVM-based algorithms). Before anomaly detection, feature selection is executed on the performance metric data to reduce data dimensionality and reserve some original performance metrics for future anomaly localization and fault diagnosis.

This article also designs an anomaly detection prototype system based on EaAD, as shown in Figure 3. In consideration of software development and system deployment, the prototype system is divided into three components, *i.e.*, *Collection & Transmission* (C&T) component, *Partition & Deployment* (P&D) component, and *Detection* (D) component.

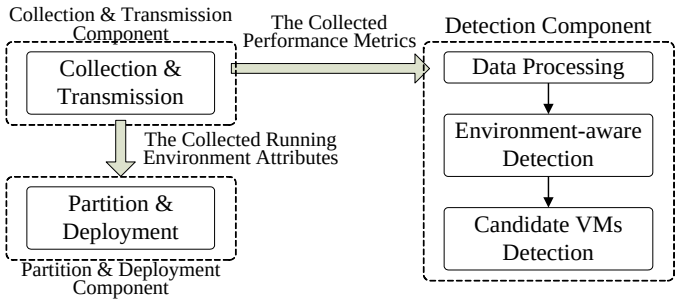


Figure 3. The Anomaly Detection Prototype System

Figure 4, shows the constructed Cloud platform. It also illustrates the physical deployment of the anomaly detection prototype system. The constructed Cloud platform comprises a management node, a number of computing clusters, and a dedicated detection cluster. The management node is responsible for the management and scheduling of Cloud platform.

Each computing cluster consists of several physical hosts. Several VMs can be deployed on each host, depending on the computing and storage capabilities of the host. Users can rent the VMs and install their applications. In the privileged domain (Dom0) of each host, a C&T component is deployed, which is mainly responsible for collecting the performance metrics and running environmental attributes data of all VMs deployed on that host, and transmitting these two categories of data to D component and P&D component respectively.

The dedicated detection cluster consists of a management server and several monitoring hosts. A P&D component is deployed in the management server, which is responsible for partitioning all of the monitored VMs into several domains. For each domain, it constructs and optimizes the monitoring network, and then assigns a dedicated detecting VM (DVM) on a selected monitoring host. A D component is then deployed in the DVM, which is responsible for processing the performance metrics data transmitted from the domain, and detecting anomalous VMs through anomaly detection algorithms.

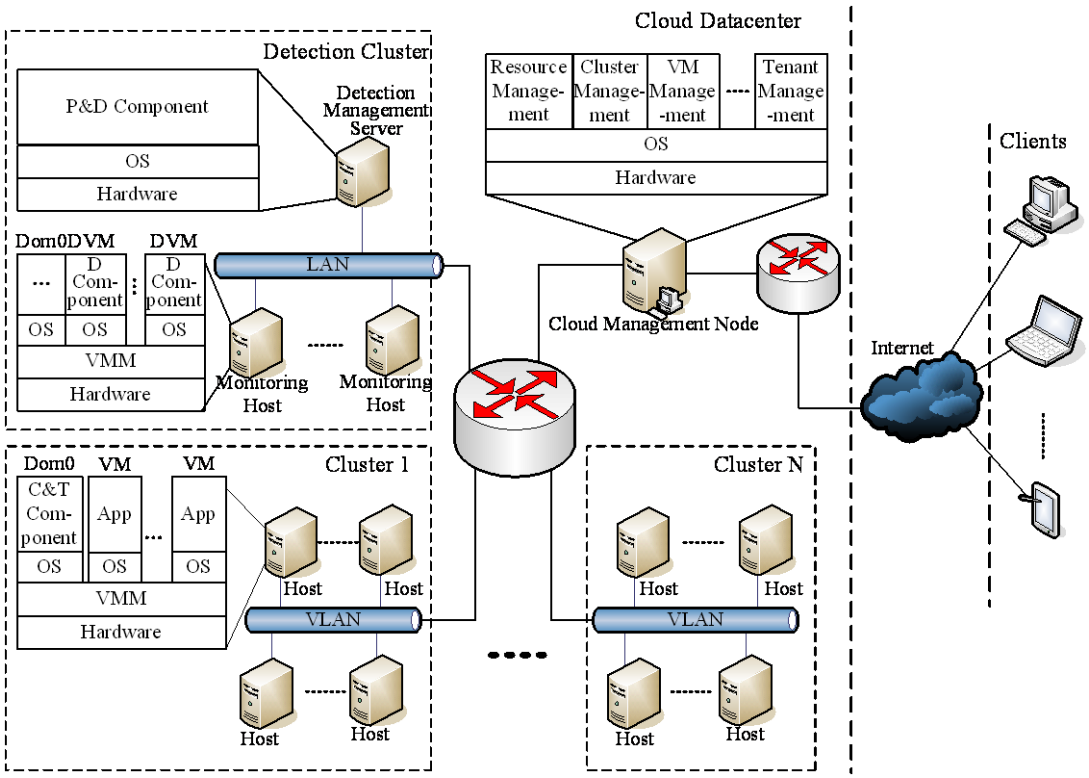


Figure 4. The Constructed Cloud Platform and the Physical Deployment of the Anomaly Detection Prototype System

4.2. The Challenges and the Equipped Algorithms

Anomaly detection of VMs in a certain monitoring domain under Cloud environment faces the following challenges.

1) *Multiple anomaly categories*. Under Cloud environment, there are many factors that may cause anomalous performance of VMs. Anomalies of VMs are diversified. Therefore, in order to further detect the types of anomalies, anomaly detection of VMs should be considered as a multi-class classification problem. However, the existing researches in literature usually only determine the states of VMs as normal or abnormal (*i.e.*, binary classification).

2) *Imbalanced training sample sets*. In general, normal samples can be easily collected. Despite frequent occurrence, anomalies are still small probability events compared with normal states. Therefore, it is not easy to collect abnormal samples. When Cloud platform is newly deployed, or a monitoring domain is newly partitioned, the training sample set only contains normal samples. After the detection framework detects abnormal states and sends to the operator for verification, abnormal samples are gradually accumulated. Therefore, a perfect anomaly detection system should be able to deal with imbalanced training sample set.

3) *Increasing number of training samples*. Since Cloud platform is a real production environment, the detection framework collects sample data of VMs in real-time. In order to accurately reflect the new trend of performance or state of VMs, the detected and verified samples should be added into the training sample set. The training of anomaly detection model usually requires much time. Therefore, the adopted anomaly detection algorithm should have the ability of online learning, *i.e.*, the detection model can be updated only according to the newly added training samples. At the same time, some selected samples should be deleted to avoid the number of training samples exceeding the capability of training sample set.

The research of Delgado *et al*. ([19]) shows that SVM is one of the best classifiers. To

cope with above challenges, this article implements several SVM-based anomaly detection algorithms in literature and equips them in EaAD. These SVM-based algorithms include:

1) Two elementary SVM-based anomaly detection algorithms, *i.e.*, two class SVM (*C*-SVM, [5][6]), one class SVM (OCSVM, [7]).
2) Multi-class SVM ([8][9]) for detecting multiple anomaly categories.
3) Imbalanced SVM ([10]) for solving imbalanced training sample sets.
4) Online learning SVM ([11][12]) for solving the problem of increasing number of training samples.

4.3. Strategies of Selecting Anomaly Detection Algorithms

According to the No Free Lunch Theorem ([20]), there is no universal detection algorithm which can solve all the problems of VM anomaly detection. Therefore, this article also designs strategies of selecting an anomaly detection algorithm from the set of equipped algorithms for different situations, which are summarized as follows.

1) If there are only normal samples, OCSVM is chosen. These situations include newly deployed Cloud platforms or newly partitioned monitoring domains. Since there is no training sample set, EaAD collects some samples and sends to the operator, there are only normal samples without abnormal ones in an initial period of running time.

2) If the ratio of one kind of samples is below a certain threshold (*e.g.*, the proportion of the number of minority class to the total number of training sample set is less than 5%), *i.e.*, the training sample set is imbalanced, imbalanced SVM is chosen. Imbalanced SVM can effectively solve the problem of imbalanced classification, thus improving the accuracy of anomaly detection.

3) If there are multiple anomaly categories, and the ratio of the number of each category exceeds a certain value, multi-class SVM is chosen. Along with the operation of Cloud platform, various anomaly samples are detected and sent to the operator for verification, thus gradually accumulating a training sample set which contains all kinds of anomalies. Then EaAD switches to multi-class SVM.

4) When EaAD stably operates for a period of time, and the number of training samples reaches a certain value (such as 30% V_T), then online learning SVM is switched. Since then EaAD still collects all kinds of samples in real time (part of the samples may be added to the training sample set to update the anomaly detection model). The incremental learning process updates the anomaly detection model with small cost, while the decremental learning process ensures that the training sample size will not exceed the capacity limit.

5. Experiments and Analyses

This section tests the performance of the equipped anomaly detection algorithms in EaAD. This article uses the following performance measures. (F_P, False Positive; F_N, False Negative; T_P, True Positive; T_N, True Negative)

1) $\text{Sensitivity} = T_P/(T_P + F_N)$.

2) $\text{Specificity} = T_N/(F_P + T_N)$.

3) $\text{Precision} = T_P/(F_P + T_P)$

4) $\text{F1-Measure} = 2PR/(P+R)$, where P is Precision, R is Recall defined as $T_P/(F_N + T_P)$ (*i.e.*, Recall is equivalent with Sensitivity).

(1) Experiments of multi-class SVM for multiple anomaly categories

In order to test the performance of multi-class SVM, four types of anomalies are

injected to randomly selected VMs under Cloud platform. These anomalies are: anomaly in computation resource consumption (ComAnomaly), anomaly in memory resource consumption (MemAnomaly), anomaly in disk I/O operation (DiskAnomaly), and anomaly in network access (NetAnomaly). A training sample set including 1000 samples is constructed, where normal samples and four kinds of abnormal samples account for 200 respectively. The detection model based on multi-class SVM is trained on this training sample set. At the same time, a testing sample set including 1000 samples is constructed. Table 1 illustrates the confusion matrix of multi-class SVM on this testing sample set, the performance measures computed based on this confusion matrix are listed in Table 2.

Table 1. The Confusion Matrix of Multi-Class SVM

detection result / real label	Normal	Com-Anomaly	Mem-Anomaly	Disk-Anomaly	Net-Anomaly
Normal	187	4	2	0	7
ComAnomaly	0	189	11	0	0
MemAnomaly	0	3	192	5	0
DiskAnomaly	4	4	3	189	0
NetAnomaly	7	7	5	2	179

Table 2. Performance Measure Results of Multi-Class SVM

Algorithm	T_P	F_P	F_N	T_N	Sensitivity	Specificity	Precision	F1-Measure
multi-class SVM	749	13	51	187	0.936	0.935	0.983	0.959

For detection of multiple kinds of anomalies, false positives (F_P) is defined as a normal state is determined as a certain kind of anomaly, while false negative (F_N) is defined as one kind of anomaly is determined as a normal state or another kind of anomaly. The experimental results in Table 2 show that the F_N value is relatively high which results in a relatively low Sensitivity. This is because that multi-class classification is usually more difficult to deal with than binary classification. Therefore, there still exists improvement space for multi-class SVM algorithm.

(2) Experiments of imbalanced SVM on imbalanced training sample sets

In order to test the performance of imbalanced SVM, a training sample set including 5000 samples is constructed, where only 50 (*i.e.*, 1%) samples are abnormal. The detection model based on imbalanced SVM is trained on this training sample set. At the same time, a testing sample set including 1000 samples is constructed. Table 3 lists the performance measure results of imbalanced SVM on this testing sample set. The results show that the performance of imbalanced SVM is not acceptable, and improvement is expected in future.

Table 3. Performance Measure Results of Imbalanced SVM

Algorithm	T_P	F_P	F_N	T_N	Sensitivity	Specificity	Precision	F1-Measure
imbalanced SVM	467	28	33	472	0.934	0.944	0.943	0.939

(3) Experiments of online learning SVM for increasing number of training samples

Finally, in order to test the performance of online learning SVM, an initial training sample set including 5000 samples is constructed. The capacity of the training sample set is set as 5200. The detection model based on online learning SVM is trained on this training sample set. At the same time, a testing sample set including 5000 samples is constructed. A sample from every 10 test samples is randomly selected as the newly added training sample to simulate the situation of increasing number of training samples.

When the number of training samples reaches the limit of capacity, a training sample is deleted each time from the training sample set during the decremental learning process. Table 4 lists performance measure results of online learning SVM on this testing sample set. Table 5 lists time efficiency experimental results in terms of the average time for updating the detection model when adding or deleting a training sample. Averagely, it takes EaAD over 150 ms to update the detection model.

Table 4. Performance Measure Results of Online Learning SVM

Algorithm	T_P	F_P	F_N	T_N	Sensitivity	Specificity	Precision	F1-Measure
online learning SVM	2383	141	117	2359	0.953	0.944	0.944	0.949

Table 5. Time Efficiency Results of Online Learning SVM

Algorithm	Adding a training sample	deleting a training sample
online learning SVM	151.7 ms	153.8 ms

From Table 4 and 5, it is concluded that online learning SVM achieves acceptable performance measures since the detection model is retrained each time when adding a new sample or deleting a selected sample; yet the time efficiency of online learning SVM is relatively low. Therefore, the emphasis of online learning SVM should be put upon the improvement of time efficiency.

6. Conclusion and Future Work

In order to improve the dependability of Cloud platform, this article proposes an environment-aware anomaly detection framework termed EaAD. The most prominent feature of EaAD is that it partitions all the VMs in Cloud platform into several monitoring domains, so as to make VMs in a same monitoring domain have similar running environment. EaAD improves the accuracy of anomaly detection due to excluding the performance deviation between VMs caused by different running environment. This article also conducts experiments on the equipped SVM-based anomaly detection algorithms and points out the improvement directions.

Currently, EaAD mainly adopts the existing techniques in such areas as feature extraction and anomaly detection. The future work of this article will focus on improving these techniques, and the detection accuracy will be expected to be further enhanced.

Acknowledgments

The authors are grateful to the editor and anonymous reviewers for their valuable comments on this article.

The work of this article is supported by National Natural Science Foundation of China (Grant No. 61272399 and No. 61572090).

References

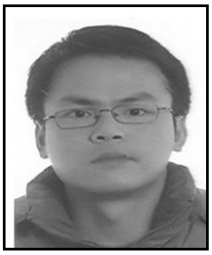
[1] R. Kumar, S. Vijayakumar, and S. A. Ahamed, “A pragmatic approach to predict hardware failures in storage systems using MPP database and big data technologies”, In: Proceedings of the 2014 IEEE International Advance Computing Conference, (**2014**), pp. 779-788.

[2] F. Langner and A. Andrzejak, “Detecting software aging in a cloud computing framework by comparing development versions”, In: Proceedings of IFIP/IEEE International Symposium on Integrated Network Management, (**2013**), pp. 896-899.

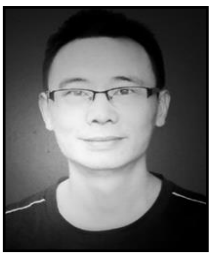
[3] S. Kikuchi and K. Hiraishi, “Improving reliability in management of cloud computing infrastructure by formal methods”, In: Proceedings of IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World, (**2014**), pp. 1-7.

[4] S. N. Brohi, M. A. Bamiah, M. N. Brohi, and R. Kamran, "Identifying and analyzing security threats to virtualized cloud computing infrastructures", In: Proceedings of International Conference on Cloud Computing Technologies, Applications and Management, (**2012**), pp. 151-155.
[5] C. Cortes and V. N. Vapnik, "Support-vector networks", Machine Learning, vol. 20, no. 3, (**1995**), pp. 273-279.
[6] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines", ACM Transactions on Intelligent Systems and Technology, vol. 2, no. 3, (**2011**), Article 27.
[7] B. Schölkopf, J. C. Platt, J. S. Taylory, A. J. Smola, and R. C. Williamson, "Estimating the support of a High-Dimensional Distribution", Neural Computation, vol. 13, (**2001**), pp. 1443-1471.
[8] J. C. Platt, N. Cristianini, and J. S. Taylory, "Large Margin DAGS for Multiclass Classification", In: Proceedings of Neural Information Processing System, (**2000**), pp. 547-553.
[9] G. P. Wang, S. Y. Chen, and J. Liu, "Anomaly-based intrusion detection using multiclass-SVM with parameters optimized by PSO", International Journal of Security and its Applications, vol. 9, no. 6, (**2015**), pp. 227-242.
[10] T. Imam, K. M. Ting, and J. Kamruzzaman, "z-SVM: An SVM for Improved Classification of Imbalanced Data", In: Proceedings of 19th Australian Joint Conference on Artificial Intelligence, (**2006**), pp. 264-273.
[11] Y. Yi, J. S. Wu, and W. Xu, "Incremental SVM based on reserved set for network intrusion detection", Expert Systems with Applications, vol. 38, no. 6, (**2011**), pp. 7698-7707.
[12] J. Zheng, F. R. Shen, H. J. Fan, and J. X. Zhao, "An online incremental learning support vector machine for large-scale data", Neural Computing & Applications, vol. 22, (**2013**), pp. 1023-1035.
[13] H. S. Pannu, J. G. Liu, Q. Guan, and S. Fu, "AFD: Adaptive Failure Detection System for Cloud Computing Infrastructures", In: Proceedings of 31st IEEE International Performance Computing and Communications Conference (IPCCC), (**2012**), pp. 71-80.
[14] Z. L. Lan, Z. M. Zheng, and Y. W. Li, "Toward automated anomaly identification in large-scale systems", IEEE Transactions on Parallel and Distributed Systems, vol. 21, no. 2, (**2010**), pp. 174-187.
[15] H. B. Mi, H. M. Wang, Y. F. Zhou, M. R. T. Lyu, and H. Cai, "Toward Fine-Grained, Unsupervised, Scalable Performance Diagnosis for Production Cloud Computing Systems", IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 6, (**2013**), pp. 1245-1255.
[16] Q. Guan, C. C. Chiu, and S. Fu, "CDA: A cloud dependability analysis framework for characterizing system dependability in cloud computing", In: Proceedings of IEEE Pacific Rim International Symposium on Dependable Computing (PRDC), (**2012**), pp. 11-20.
[17] C. W. Wang, W. Talwar, K. Schwan, and P. Ranganathan, "Online detection of utility cloud anomalies using metric distributions", In: Proceedings of IEEE/IFIP Network Operations and Management Symposium, (**2010**), pp. 96-103.
[18] S. C. Meng, L. Liu, and T. Wang, "State Monitoring in Cloud Datacenters", IEEE Transactions on Knowledge and Data Engineering, vol. 23, no. 9, (**2011**), pp. 1328-1344.
[19] M. F. Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?", Journal of Machine Learning Research, vol. 15, (**2014**), pp. 3133-3181.
[20] R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern Classification", John Wiley & Sons Press, (**2001**).

Authors



GuiPing Wang, received his B.S. degree, M.S. degree, and Ph.D. degree at Chongqing University, P. R. China, in 2000, 2003, and 2015, respectively. Currently he is a lecturer in College of Information Science and Engineering, Chongqing Jiaotong University. His research interests include machine learning, dependability analysis and design of distributed systems, and cloud computing. As the first author, he has published about 20 articles in related research areas during recent years at journals including IPL, Optik, and ESWA.



JiaWei Wang, received his M.S. degree at Chongqing University, P. R. China, in 2006. Currently he is an associate professor in College of Information Science and Engineering, Chongqing Jiaotong University. His research interests include big data analytics, intelligent computing, software engineering, *etc*.