International Journal of Hybrid Information Technology
Vol.9, No.2 (2016), pp. 439-450
http://dx.doi.org/10.14257/ijhit.2016.9.2.39

Data-driven Anomaly Detection Method for Monitoring Runtime Performance of Cloud Computing Platforms

Mingwei Lin[1], Zhiqiang Yao[1*], Fei Gao[1] and Yang Li[1]

[1]Faculty of Software, Fujian Normal University, Fuzhou 350108, Fujian, China
yzqfnu@163.com

Abstract

Cloud computing platforms are complex system, which consist of a lot of software working together. Because of software defects, cloud computing platforms may has performance anomaly during runtime. In this paper, a data-driven anomaly detection method is proposed to monitor runtime performance for cloud computing platforms. The proposed method can not only detect the performance anomaly of cloud computing platforms during runtime, but also find out which performance metric results in the anomaly. A series of experiments are conducted on a real private cloud computing platform based on OpenStack and experimental results show the proposed method is better than previous anomaly detection methods for cloud computing platforms.

Keywords*: Data-driven, Anomaly detection, Cloud computing, Local outlier factor*

1. Introduction

Cloud computing, which is a kind of new computing method based on Internet, has been the research focus in both industry and academia [1]. Many large-scale IT companies have their own cloud computing platforms. For example, Amazon releases the Elastic Compute Cloud (EC2) and Simple Storage Service (S3) [2], Microsoft also customizes an operating system called Microsoft Azure for cloud computing platforms [3]. Experts and scholars from academia follow IT companies and have developed a series of open source software for cloud computing. For instance, the company Eucalyptus Systems releases a free and open source computer software, which is called Eucalyptus (Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems) and used for building Amazon Web Services (AWS)-compatible private and hybrid cloud computing platforms [4]. Rajiv Ranjan from University of Melbourne also develops an extensible simulation toolkit called CloudSim [5]. The open source CloudSim simulation toolkit is widely used to model and simulate cloud computing platforms and evaluate resource provisioning algorithms by researchers. These research achievements promote the quick development of cloud computing. Currently, cloud computing is widely used in many fields such as medical treatment, education, agriculture, and manufacture.

As the market share of cloud computing grows quickly, a large number of countries list cloud computing as the national overall development strategy and then promote the industrial development of cloud computing. For example, United States government releases a Federal Cloud Computing Strategy white paper in February 2011. This white paper said that 25% of IT budget would be used to migrate current IT systems to cloud computing platform in order to solve the problems that the utilization of e-government infrastructure is very low. United Kingdom government also published a G-Cloud plan in November 2011. This G-Cloud plan invests 6000 million pounds to build public cloud service network in order to reduce the IT expense of government.

In the traditional IT infrastructure, each business system monopolizes computing, storage and network resources, but the average utilization of hardware resources for business systems is less than 10% [6]. Moreover, hardware devices are changed frequently. Typically, they are updated every three to five years. In this case, a number of hardware resources are wasted and IT operation costs of enterprises are very high. Moreover, in the traditional IT infrastructure, it takes a long time to let new business systems online and results in delaying the online time of business systems seriously.

Cloud computing often uses the virtualization technology to consolidate existing hardware resources and forms the shared resource pool [7]. In this case, business systems built on cloud computing platforms can obtain computing, storage, and network resources on demand. It can effectively solve the problems of traditional IT infrastructure. As cloud computing develops and matures continuously, more and more enterprises deploy their business systems to cloud computing platforms in order to improve the utilization of hardware resources and reduce the IT operation cost. As the number of business systems deployed on cloud computing platforms increases continuously, the scales of cloud computing platforms enlarge continuously and the cloud computing platforms become increasingly complex. Moreover, virtual machines on the cloud computing platforms share hardware resources and all software has software defects, especially for complex system such as cloud computing platforms [8]. All of these cause that cloud computing platforms are prone to performance anomaly during runtime.

In order to avoid the performance anomaly of cloud computing platforms, a large number of performance anomaly detection methods have been proposed. However, previous anomaly detection methods focus on detect the performance anomaly of cloud computing platforms. None of them do research on how to find out performance metrics that incur the performance anomaly. In this paper, an efficient data-driven anomaly detection method is proposed to monitor the performance anomaly of cloud computing platforms during their runtime. The proposed anomaly detection method introduces an improved local outlier factor algorithm to detect the performance anomaly and then finds out the performance metrics that incur the performance anomaly. A series of experiments are conducted on a private cloud computing platform that is built by using OpenStack and Xen software. Experimental results show that the proposed anomaly detection method outperforms previous anomaly detection methods for cloud computing platforms.

The rest of this paper is organized as follows. Section 2 briefly reviews related works. Our proposed anomaly detection method is presented in Section 3 in detail. Section 4 shows the performance evaluation. Finally, conclusions are drawn in Section 5.

2. Related Works

In this Section, commonly used anomaly detection techniques are first described and then existing works on performance anomaly detection for cloud computing platforms are briefly reviewed.

2.1. Anomaly Detection Techniques

Anomaly detection is a process that detects an anomalous data instance from a dataset [9]. The anomalous data instances are also referred to as anomaly. Anomaly is usually categorized into three types, which are point anomaly, contextual anomaly, and collective anomaly [10]. If an individual data instance can be considered as anomalous with respect to the rest data, then the data instance is referred to as a point anomaly. If a data instance is anomalous in a specific context, but not otherwise, then it is defined as a contextual anomaly. If a collection of related data instances is anomalous with respect to the entire dataset, then it is considered as a collective anomaly. The performance anomaly of cloud computing platforms belongs to the type of point anomaly [11]. Therefore, this paper briefly reviews the anomaly detection techniques that are designed for detecting point anomaly.

(1) Statistical anomaly detection techniques

The underlying design principle of any statistical anomaly detection technique is that an anomaly is a data instance which is suspected of being partially or wholly irrelevant because it is not generated by the stochastic model assumed [12]. This kind of anomaly detection techniques assume that normal data instances often appear in high probability regions of a stochastic model assumed and anomalies often occur in the low probability regions of the stochastic model. Statistical anomaly detection techniques fit a statistical model (usually for normal data instances) to the given dataset and then use a statistical inference test to determine whether an unknown data instance belongs to this statistical model or not. If data instances have a low probability of being generated from a fitted statistical model based on the applied test statistic, they are considered as anomalies. If the assumptions regarding the underlying data distribution hold true, statistical anomaly detection techniques could identify anomalies from a dataset effectively. However, the statistical anomaly detection techniques rely on the assumptions that the data instances are generated from a particular distribution and the assumption usually do not hold true, especially for high dimensional real datasets.

(2) Clustering-based anomaly detection techniques

Clustering is a technique that group similar data instances into several clusters [13]. Clustering-based anomaly detection techniques introduce the underlying principle of clustering technique and they often use the Euclidean distance to measure the similarity between two data instances [14]. Clustering-based anomaly detection techniques are usually categorized into three types.

The first type of clustering-based anomaly detection techniques assumes that normal data instances belong to a cluster in the data and anomalies do not belong to any cluster [15]. Clustering-based anomaly detection techniques based on this assumption applies a known clustering-based algorithm to the given data in order to form clusters and defines any data instance that does not belong to any cluster as anomalous.

The second type of clustering-based anomaly detection techniques is designed based on the assumption that normal data instances are located close to the centroid of their closest cluster and anomalies are far away from the centroid of their closest cluster [16]. This type of clustering-based anomaly detection techniques is composed of two steps. In the first step, a clustering algorithm is applied to group the given data instances into clusters. In the second step, the distance of each data instance to the centroid of its closest cluster is computed as its anomaly score.

The third type of clustering-based anomaly detection techniques assumes that normal data instances belong to large and dense clusters and anomalies belong to either small clusters or sparse clusters [17]. Clustering-based anomaly detection techniques based on this assumption consider data instances belonging to clusters whose size and/or density is below a threshold, as anomalous.

Clustering-based anomaly detection techniques do not need to know the distribution of data as statistical anomaly detection techniques. However, clustering-based anomaly detection techniques must choose a suitable parameter for the width of clusters.

(3) Classification-based anomaly detection techniques

Classification-based anomaly detection techniques is composed of two steps [18]. In the first step (training phase), a classifier is learned using a set of labeled training data. In the second step (testing phase), the classifier is adopted to classify a test instance as normal or anomalous. Classification-based anomaly detection techniques assume that a classifier that can distinguish normal class from anomalous class can be learned using a set of labeled training data. Based on the labels of training data in the training phase, classification-based anomaly detection techniques can be categorized into two types: multi-class and one-class anomaly detection techniques.

The multi-class classification-based anomaly detection techniques operate based on the assumption that the labeled training data belonging to multiple normal classes and the one-class classification-based anomaly detection techniques assume that all labeled training data belong to only one class. The classification-based anomaly detection techniques, especially the multi-class ones, can distinguish testing data instances belonging to different classes. However, the classification-based anomaly detection techniques depend on the availability of accurate labels for various normal classes, which is usually impossible.

(4) Nearest neighbor-based anomaly detection techniques

The nearest neighbor-based anomaly detection techniques are designed based on the assumption that normal data instances appear in the dense neighborhoods and anomalies occur far from their closest neighbors [19]. Such techniques define a distance between two data instances, which are often computed using the Euclidean distance. They can be categorized into two types. The first type of nearest neighbor-based anomaly detection techniques define the anomaly score of a data instance as its distance to its k^{th} nearest neighbor and the second type of nearest neighbor-based anomaly detection techniques compute the anomaly score of a data instance as its relative density. Nearest neighbor-based anomaly detection techniques are unsupervised in nature and do not require labels for various classes. Moreover, they do not need to make any assumptions regarding the distribution for the data. However, if the anomalies within the data have enough close neighbors, the nearest neighbor-based anomaly detection techniques will fail to classify them correctly.

2.2. Existing Works on Anomaly Detection for Cloud Computing Platforms

Wang, *et al.,* proposed an online EbAT (Entropy-based Anomaly Testing) approach to detect anomaly for cloud computing platforms [20]. The EbAT detects anomalies by analyzing the data performance metric data distributions rather than individual metric data thresholds. The entropy is used as a measurement to capture the degree of dispersal or concentration of such distributions. Experimental results show that the EbAT approach outperforms threshold-based approaches.

Pannu, *et al.,* proposed a self-evolving anomaly detection framework for developing highly dependable utility clouds [21]. The self-evolving anomaly detection framework contains two components. The first one is anomaly detector determination. For a new performance metric data record from cloud computing platforms, the anomaly detector computes its abnormality score. If the abnormality score is larger than a threshold, a warning is triggered with the type of abnormality, which can help the administrators of cloud computing platforms to pinpoint the anomaly. The second one is anomaly detector evolving and working dataset selection. The anomaly detector can self-evolve by learning from newly verified and labeled data records in order to improve the performance of anomaly detector.

Bhaduri, *et al.,* proposed an automated fault detection framework for cloud systems, which is called FDCS and uses the Ganglia system to collect the performance metric data from each machine in the cloud computing platforms [22]. The FDCS uses the Euclidean distance to compute the anomaly score of each machine and then identify whether the machine is faulty or not.

Smith, *et al.,* showed an autonomic anomaly detection framework for large-scale computing cloud systems [23]. The proposed autonomic anomaly detection framework first employs the data transformation step to tackle data diversity. Then, the feature selection step is conducted to reduce the data dimension for quick and better analysis. Finally, the anomaly detection step identifies the nodes that show significantly different performance metric data from others as faulty nodes.

Nguyen, *et al.,* proposed a black-box online fault localization system called FChain for cloud computing systems to pinpoint faulty components [24]. The FChain is composed of two parts: FChain master and FChain slave. The FChain slave collects the performance metric data from virtual machines continuously and normal workload fluctuation patterns are learned to detect SLO violations. When a SLO violation is detected, the FChain master pinpoints the faulty virtual machines.

Guan, *et al.,* proposed an adaptive anomaly identification mechanism for cloud computing infrastructures [25]. The proposed adaptive anomaly identification mechanism first explores the most relevant principal components for each type of possible failures. Then, it leverages an adaptive Kalman filter to identify anomalies. Finally, it recursively learns from the newly verified detection results in order to refine the anomaly detection performance. Experimental results evaluate the effectiveness of the proposed adaptive anomaly identification mechanism.

3. Data-Driven Anomaly Detection Method

Through analysis of existing works on anomaly detection for cloud computing platforms, we can found that none of them focus on pinpointing the performance metrics that result in the anomalies of cloud computing platforms. In order to solve this problem, an efficient data-driven anomaly detection method is proposed to identify the anomalies and then pinpoint the performance metrics that lead to the anomalies.

3.1. Identify Anomalies

The data-driven anomaly detection method introduces the local outlier factor algorithm to identify anomalies for cloud computing platforms. The local outlier factor algorithm is a kind of nearest neighbor-based anomaly detection techniques, which was first proposed by Breunig, *et al.*, in 2000 [26]. The local outlier factor algorithm assigns to each data instance a degree of being an anomaly. This degree is called the local outlier factor (LOF), which indicates that whether the data instance is located in a dense region or not. The local outlier factor of each data instance can be computed as follows.

(1) Computing the *k*-distance of a data instance *p*

For any positive integer *k*, if it satisfies the following two conditions, the *k*-distance of a data instance *p*, denoted as $k-distance(p)$, is defined as the distance $d(p,q)$ between the data instance *p* and an data instance *q*.

(1.1) For at least *k* data instances $q^{'} \in D \setminus \{p\}$ it holds that $d(p,q^{'}) \le d(p,q)$;

(1.2) For at most k-1 data instances $q^{'} \in D \setminus \{p\}$ it holds that $d(p,q^{'}) < d(p,q)$.

The distance $d(p,q)$ between two data instances is usually computed using the Euclidean distance between these two data instances.

(2) Determining the *k*-distance neighborhood of the data instance *p*

Given the *k*-distance of data instance *p*, the *k*-distance neighborhood of data instance *p* is a set of data instances whose distances from data instance *p* are not larger than the *k*-distance of data instance p.

$$KNN(p) = N_k(p) = \{q \in D \setminus \{p\} \mid d(p,q) \le k-distance(p)\} \quad (1)$$

where the term $KNN(p)$ is the *k*-distance neighborhood of data instance *p*.

(3) Calculating the reachability distance of the data instance *p* w.r.t a data instance *q*

The reachability distance of the data instance *p* with respect to a data instance *q* is defined as

$$reach-dist_k(p,q) = \max\{k-distance(q), d(p,q)\} \quad (2)$$

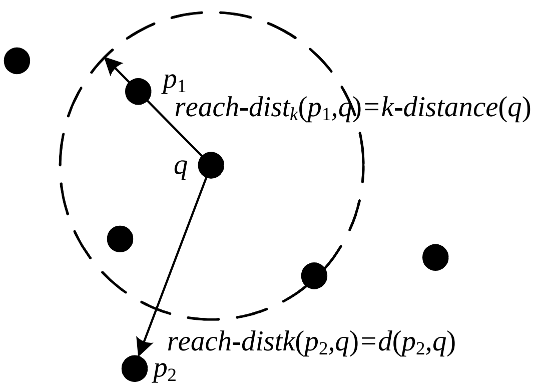
p_1
$reach\text{-}dist_k(p_1,q)=k\text{-}distance(q)$
q
$reach\text{-}distk(p_2,q)=d(p_2,q)$
p_2

Figure 1. Reachability Distances of the Data Instances p_1 and p_2 w.r.t the Data Instance q for $k=3$

Figure 1 shows the reachability distance of the data instances p_1 and p_2 with respect to the data instance *q* when *k* is equal to 3. Because the data instance p_1 appears in the *k*-distance neighborhood of the data instance *q*, the reachability distance of the data instance p_1 with respect to the data instance *q* is the *k*-distance of the data instance *q*. The data instance p_2 is not inside the *k*-distance neighborhood of the data instance *q*, the reachability distance of the data instance p_2 with respect to the data instance *q* is equal to the actual distance between the two data instances.

(4) Calculating the local reachability density of the data instance *p*

The local reachability density of the data instance *p* is defined as

$$lrd_k(p) = 1 \Bigg/ \left[\frac{\sum_{q \in N_k(p)} reach-dist_k(p,q)}{|N_k(p)|} \right] \tag{3}$$

Intuitively, the local reachability density of the data instance *p* is computed as the inverse of the average reachability distance of the data instance *p* with respect to all the data instances within the *k*-distance neighborhood of the data instance *p*.

(5) Calculating the local outlier factor (lof) of the data instance *p*

The local outlier factor of the data instance *p* is defined as

$$LOF_k(p) = \frac{\sum_{q \in N_k(p)} \frac{lrd_k(q)}{lrd_k(p)}}{|N_k(p)|} \tag{4}$$

Intuitively, the local outlier factor of the data instance is calculated as the average of the ratio of the local reachability density of the data instance *p* and those of each data instance in the k-distance neighborhood of the data instance *p*.

If the local outlier factor of a data instance is approximately equal to 1, then it is identified as a normal data instance. If the local outlier factor a data instance is much higher than 1, then it is considered as an anomaly.

In order to detect the performance anomaly of cloud computing platforms, the performance metric data are collected from cloud computing platforms during runtime continuously and then the local outlier factor of each performance metric data is calculated using the above local outlier factor algorithm. If the local outlier factor of current performance metric data is approximately equal to 1, it indicates that current cloud computing platforms perform well. If the local outlier factor of current performance metric data is much higher than 1, it indicates that current cloud computing platforms are very likely to have a performance anomaly.

3.2. Pinpoint the Anomalous Performance Metrics

When a performance anomaly is identified from cloud computing platforms during runtime, the data-driven anomaly detection method continues to pinpoint the anomalous performance metrics to help system administrators to adopt some measures.

The data-driven anomaly detection method analyzes the formula of the local outlier factor and looks for anomalous performance metrics that contribute to the anomalous performance metric data significantly.

(1) The formula of the local outlier factor is analyzed as follows.

$$\begin{aligned} LOF_k(p) &= \frac{\sum_{q\in N_k(p)} \frac{lrd_k(q)}{lrd_k(p)}}{|N_k(p)|} \\ &= \frac{1}{|N_k(p)|}\sum_{q\in N_k(p)} \frac{lrd_k(q)}{lrd_k(p)} \\ &= \frac{1}{|N_k(p)|}\sum_{q\in N_k(p)} lrd_k(q) \frac{\sum_{q\in N_k(p)} reach-dist_k(p,q)}{|N_k(p)|} \\ &= \frac{\sum_{q\in N_k(p)} lrd_k(q) \sum_{q\in N_k(p)} \max\{k-distance(q), d(p,q)\}}{|N_k(p)|^2} \end{aligned} \tag{5}$$

where the term $LOF_k(p)$ is the local outlier factor of the data instance *p* and $N_k(p)$ is its *k*-distance neighborhood.

If the *k*-distance of the data instance *q*, denoted as $k-distance(q)$, is not less than the distance between the data instances *p* and *q*, then the local outlier factor of the data instance *p* is computed as

$$\begin{aligned} LOF_k(p) &= \frac{\sum_{q\in N_k(p)} lrd_k(q) \sum_{q\in N_k(p)} k-distance(q)}{|N_k(p)|^2} \\ &= \frac{\sum_{q\in N_k(p)} lrd_k(q) \sum_{q\in N_k(p)} (q-o_q)^2}{|N_k(p)|^2} \end{aligned} \tag{6}$$

where the term o_q is the *k*th-nearest neighbors of the data instance *q*.

Let us assume that the performance metric data of the data instance $q=\left[q^1,q^2,\cdots,q^m\right]^T$ and the performance metric data of the data instance $o_q=\left[o_q^1,o_q^2,\cdots,o_q^m\right]^T$.

where the term *m* represents the number of performance metrics.

Then, the local outlier factor of the data instance *p* can be transformed as follows.

$$
\begin{aligned}
&LOF_k(p) \\
&= \frac{\sum_{q\in N_k(p)} lrd_k(q) \sum_{q\in N_k(p)} \left(q-o_q\right)^2}{\left|N_k(p)\right|^2} \\
&= \frac{\sum_{q\in N_k(p)} lrd_k(q) \sum_{q\in N_k(p)} \left[\left(q^1-o_q^1\right)^2+\left(q^2-o_q^2\right)^2+\cdots+\left(q^m-o_q^m\right)^2+\left(q^m-o_q^m\right)^2\right]}{\left|N_k(p)\right|^2} \\
&= \frac{\sum_{q\in N_k(p)} lrd_k(q) \left[\sum_{q\in N_k(p)} \left(q^1-o_q^1\right)^2+\sum_{q\in N_k(p)}\left(q^2-o_q^2\right)^2+\cdots+\sum_{q\in N_k(p)}\left(q^m-o_q^m\right)^2\right]}{\left|N_k(p)\right|^2} \qquad (7)\\
&= \frac{\sum_{q\in N_k(p)} lrd_k(q) \sum_{q\in N_k(p)} \left(q^1-o_q^1\right)^2}{\left|N_k(p)\right|^2}+\frac{\sum_{q\in N_k(p)} lrd_k(q) \sum_{q\in N_k(p)} \left(q^2-o_q^2\right)^2}{\left|N_k(p)\right|^2}+\cdots \\
&\quad+\frac{\sum_{q\in N_k(p)} lrd_k(q) \sum_{q\in N_k(p)} \left(q^m-o_q^m\right)^2}{\left|N_k(p)\right|^2} \\
&= LOF_k\left(p^1\right)+LOF_k\left(p^2\right)+\cdots+LOF_k\left(p^m\right)
\end{aligned}
$$

where the term $\left[p^1, p^2,\cdots, p^m\right]^T$ represents the performance metric data of data instance *p*.

Finally, the contribution of each performance metric is computed as follows.

$$contribution\left(p^i\right)=\frac{LOF_k\left(p^i\right)}{LOF_k(p)} \qquad (8)$$

where the term *i* is the *i*th performance metric.

When the *k*-distance of the data instance *q*, denoted as $k-distance(q)$, is less than the distance between the data instances *p* and *q*, then the local outlier factor of the data instance *p* can also be transformed to be the equation (7).

The performance metrics are sorted based on their contributions. The performance metric with the highest contribution is identified as anomalous and then the anomalous performance metric can help the system administrators seek out the potential problems that result in the performance anomaly of cloud computing platforms.

4. Performance Evaluation

In this section, the effectiveness of the proposed data-driven anomaly detection method is evaluated.

4.1. Experiment Setup

A series of experiments are conducted on a private cloud computing platform that is built using the open source software OpenStack and Xen. This private cloud computing platform is composed of six physical nodes, which are connected with a Gigabit Ethernet. One of them is chosen as a cloud controller, while the rest physical nodes are selected as compute nodes. Xen is used to created virtual machines. Figure 2 shows the experimental environment of the private cloud computing platforms.

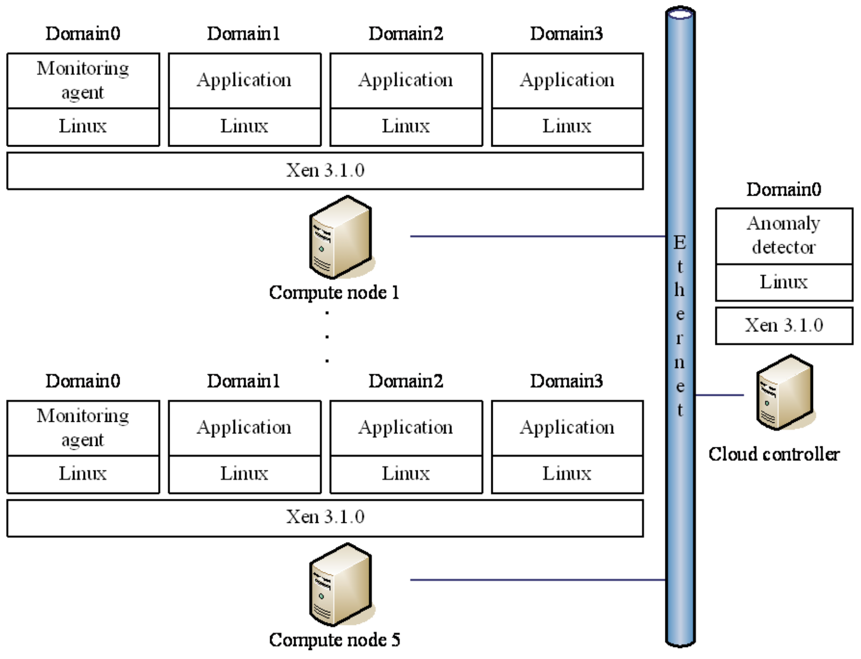

Figure 2. Experiment Environment

As presented in Figure 2, the data-driven anomaly detection method consists of a monitoring agent and an anomaly detector, respectively. The monitoring agent is in charge of collecting the performance metric data from cloud computing platforms continuously, and the anomaly detector is responsible for detecting the anomalous performance metric data and pinpointing the anomalous performance metric that results in the performance anomaly.

In order to generate real workloads, the distributed online service benchmark application RUBIS is built on the private cloud computing platform and then a fault injection application is introduced to inject 50 faults with four types into the virtual machines, such as MemLeak, CPUHog, DiskHog, and NetHog.

The precision, recall, and F-measure are used to evaluate the performance of the proposed data-driven anomaly detection method. They are defined as follows.

$$Precision = \frac{N_{TP}}{N_{TP} + N_{FP}} \tag{10}$$

$$Recall = \frac{N_{TP}}{N_{TP} + N_{FN}} \tag{11}$$

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{12}$$

where the term N_{TP} is the number of anomalous performance metric data that are identified as anomalous, the term N_{FP} represents the number of normal performance metric data, which are identified as anomalous. The term N_{FN} is the number of anomalous performance metric data that are identified as normal.

In order to assess the performance of the proposed data-driven anomaly detection method, the proposed data-driven anomaly detection method is compared with previous methods that are proposed for cloud computing platforms, which are the EbAT, FDCS, and FChain.

4.2. Experimental Results

In this section, experimental results are presented and analyzed.

Table 1. Precision, Recall, and F-Measure for Different Anomaly Detection Methods

Anomaly detection methods	N_{TP}+N_{FP}	N_{TP}	**Precision**	**Recall**	**F-measure**
Proposed data-driven anomaly detection method	51	48	94.12%	96%	95.05%
EbAT	55	47	89.09%	94%	91.48%
FDCS	53	46	86.79%	92%	89.32%
FChain	52	42	80.77%	84%	82.35%

Table 1 shows the precision, recall, and F-measure for four anomaly detection methods. It can be seen that the proposed data-driven anomaly detection method outperforms the EbAT, FDCS, and FChain methods in terms of precision, recall, and F-measure. The EbAT, FDCS, FChain anomaly detection methods identify the normal performance metric data fluctuation as anomalies, while the proposed data-driven anomaly detection method introduces the local outlier factor algorithm to detect the performance anomaly and then can avoid that the normal performance metric data fluctuation as anomalies. Hence, the proposed data-driven anomaly achieves the higher precision, recall, and F-measure than the EbAT, FDCS, FChain anomaly detection methods.

5. Conclusions

In this paper, an efficient data-driven anomaly detection method is proposed for real cloud computing platforms that are built using the OpenStack and Xen. The proposed data-driven anomaly detection method applies the local outlier factor algorithm to the performance metric data collected from cloud computing platforms during runtime in order to detect the performance anomaly. The local outlier factor algorithm could avoid that the normal performance metric data fluctuation as anomalies. In order to pinpoint the anomalous performance metrics that lead to the anomalous performance metric data, the proposed data-driven anomaly detection method analyzes the formula of the local outlier factor and then computes the contribution of each performance metric. A series of experiments are conducted a private cloud computing platform that is built using the open source software OpenStack and Xen. Experimental results show that the proposed data-driven anomaly method is better than previous anomaly detection methods that are designed for cloud computing platforms in terms of precision, recall, and F-measure.

Acknowledgments

The work of this paper is supported by the National Natural Science Foundation of China under Grant No. 61502102, No. 61370078, and No. 61402109, Fujian Province Education Scientific Research Projects for Young and Middle-aged Teachers under Grant No. JA15122, National Undergraduate Training Programs for Innovation and Entrepreneurship under Grant No. 201510394021, and Fujian Normal University Undergraduate Training Programs for Innovation and Entrepreneurship under Grant No. cxxl-2015163.

References

[1] F. Nadeem and R. Qaiser, “An early evaluation and comparison of three private cloud computing software platforms”, Journal of Computer Science and Technology, vol. 30, no. 3, **(2015)**, pp. 639-654.

[2] M. Balduzzi, J. Zaddach, D. Balzarotti, E. Kirda and S. Loureiro, “A security analysis of amazon's elastic compute cloud service”, Proceedings of the ACM Symposium on Applied Computing, **(2012)**, pp. 1427-1434.

[3] B. Di Martino, G. Cretella, A. Espostito and R. G. Sperandeo, “Semantic representation of cloud services: A case study for Microsoft windows azure”, Proceedings of 2014 International Conference on Intelligent Networking and Collaborative Systems, **(2015)**, pp. 647-652.

[4] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff and D. Zagorodnov, “The eucalyptus open-source cloud-computing system”, Proceedings of 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, (**2009**), pp. 124-131.

[5] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, “CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms”, Software - Practice and Experience, vol. 41, no. 1, **(2011)**, pp. 23-50.

[6] C.-F. Wu, “Learning attitude and its effect on applying cloud computing service to IT education”, International Journal of u- and e- Service, Science and Technology, vol. 6, no. 1, **(2013)**, pp. 39-48.

[7] W. Bai and W. Geng, “Research on operation management under the environment of cloud computing data center”, International Journal of Database Theory and Application, vol. 8, no. 2, **(2015)**, pp. 185-192.

[8] M. Gupta, A. B. Sharma, H. Chen and G. Jiang, “Context-aware time series anomaly detection for complex systems”, 2013 Workshop on Data Mining for Service and Maintenance, **(2013)**, pp. 14-22.

[9] M. Xie, S. Han, B. Tian and S. Parvin, “Anomaly detection in wireless sensor networks: A survey”, Journal of Network and Computer Applications, vol. 34, no. 4, **(2011)**, pp. 1302-1325.

[10] V. Chandola, A. Banerjee and V. Kumar, “Anomaly detection: A survey”, ACM Computing Surveys, vol. 41, no. 3, **(2009)**, Article no. 15.

[11] S. Fu, “Performance metric selection for autonomic anomaly detection on cloud computing systems”, IEEE Global Telecommunications Conference, **(2011)**, pp. 1-5.

[12] Y.-J. Chang, K.-P. Lin, L.-D. Chou, S.-F. Chen and T.-S. Ma, “Statistical anomaly detection for individuals with cognitive impairments”, IEEE Journal of Biomedical and Health Informatics, vol. 18, no. 1, **(2014)**, pp. 384-390.

[13] F. Ren, L. Hu, K. Zhao, H. Liang and W. Ren, “ADIC: an anomaly detection algorithm using incremental clustering”, Journal of Information and Computational Science, vol. 6, no. 2, **(2009)**, pp. 1051-1057.

[14] M. Leng, X. Lai and Y. Liu, “Clustering model for anomaly detection in time series sets”, Journal of Information and Computational Science, vol. 7, no. 1, **(2010)**, pp. 85-93.

[15] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise”, Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining, **(1996)**, pp. 226-231.

[16] Z. He, X. Xu and S. Deng, “Discovering cluster-based local outliers”, Pattern Recognition Letters, vol. 24, no. 9-10, **(2003)**, pp. 1641-1650.

[17] M. Otey, S. Parthasarathy, A. Ghoting, G. Li, S. Narravula and D. Panda, “Towards NIC-based intrusion detection”, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, **(2003)**, pp. 723-728.

[18] P.-N. Tan, M. Steinbach and V. Kumar, “Introduction to data mining”, Addison-Wesley, **(2005)**.

[19] S. Boriah, V. Chandola and V. Kumar, “Similarity measures for categorical data: A comparative evaluation”, Proceedings of the 8th SIAM International Conference on Data Mining, **(2008)**, pp. 243-254.

[20] C. Wang, V. Talwar, K. Schwan and P. Ranganathan, “Online detection of utility cloud anomalies using metric distributions”, Proceedings of the 2010 IEEE/IFIP Network Operations and Management Symposium, **(2010)**, pp. 96-103.

[21] H.-S. Pannu, J. Liu and S. Fu, “A self-evolving anomaly detection framework for developing highly dependable utility clouds”, 2012 IEEE Global Communications Conference, **(2012)**, pp. 1605-1610.

[22] K. Bhaduri, K. Das and B. L. Matthews, “Detecting abnormal machine characteristics in cloud infrastructures”, Proceedings of 11th IEEE International Conference on Data Mining Workshops, **(2011)**, pp. 137-144.

[23] D. Smith, Q. Guan and S. Fu, “An anomaly detection framework for autonomic management of compute cloud systems”, Proceedings of 34th Annual IEEE International Computer Software and Applications Conference Workshops, **(2010)**, pp. 376-381.

[24] H. Nguyen, Z. Shen, Y. Tan and X. Gu, “FChain: Toward black-box online fault localization for cloud systems”, Proceedings of 2013 IEEE 33rd International Conference on Distributed Computing Systems, **(2013)**, pp. 21-30.

[25] Q. Guan and S. Fu, "Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures", Proceedings of 2013 IEEE 32nd International Symposium on Reliable Distributed Systems, **(2013)**, pp. 205-214.
[26] M.-M. Breunig, H.-P. Kriegel, R.-T. Ng and J. Sander, "LOF: identifying density-based local outliers", Proceedings of the ACM SIGMOD International Conference on Management of Data, vol. 29, no. 2, **(2000)**, pp. 93-104.

Authors

Mingwei Lin, He received his B.S. and Ph.D. degrees from Chongqing University, China, in July 2009 and December 2014. Currently, he is a lecturer in the Faculty of Software, Fujian Normal University, China. His research interests include anomaly detection, NAND flash memory, Linux operating system, and cloud computing. He got the CSC-IBM Chinese Excellent Student Scholarship in 2012.

Zhiqiang Yao, He received the PhD degree from Xidian University, China, in 2014. Currently, he is a professor in the Faculty of Software, Fujian Normal University, ACM Professional Membership, Senior Member of China Computer Federation (CCF). His current research interests mainly focus on security in cloud computing, multimedia security.

Fei Gao, He is a junior student in the Faculty of Software, Fujian Normal University, Fuzhou, China. He majors in Software Engineering. He has applied for a National Undergraduate Training Program for Innovation and Entrepreneurship successfully. His current research interests include cloud computing, android application development, and flash memory.

Yang Li, He is a junior student in the Faculty of Software, Fujian Normal University, Fuzhou, China. Her major is Software Engineering. She is hosting a Fujian Normal University Undergraduate Training Program for Innovation and Entrepreneurship. Her current research interests include anomaly detection, cloud computing, and flash memory.