A Fault Detection and Diagnosis Approach for Multi-tier Application in Cloud Computing

Khiet Thanh Bui, Len Van Vo, Canh Minh Nguyen, Tran Vu Pham, and Hung Cong Tran

Abstract: **Ensuring the availability of cloud computing services always concerns both service providers and end users. Therefore, the system always needs precautions for unexpected cases. Accordingly, cloud computing services must be capable of identifying faults and behaving appropriately when it is abnormal to ensure the smoothness as well as the service quality. In this study, we propose a fault detection method for multi-tier web application in cloud computing deployment environment based on the Fuzzy One-class support vector machine and Exponentially Weighted Moving Average method. And then, the suspicious metrics are located by using feature selection method which based on Random Forest algorithm. To evaluate our approach, a multi-tier application is deployed by a transnational web e-Commerce benchmark by using TPC-W (TPC Benchmark™ W, simulates the activities of a business oriented transaction web server in a controlled internet commerce environment) in private cloud and then it is injected typical faults. The effectiveness of the fault detection and diagnosis are demonstrated in experiment results.**

Index Terms: **Cloud computing, fault detection, fuzzy One-class SVM, multi-tier web application**

I. INTRODUCTION

CLOUD computing has been widely used in a variety of fields like business, high performance computing, social network and scientific computing thanks to its capability to offer information technology infrastructure and application as scalable services. The cloud infrastructure contains both physical layer and abstraction layer. The physical layer typically includes server, storage and network components that are necessary to support the cloud services. The abstraction layer consists software which are deployed across the physical layer. Corresponding to cloud service models, abstraction layer is divided into three types of service models including IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Services). IaaS provides consumers with computing resource as virtual machine while PaaS offers programming language, libraries, services and tools; and applications running on cloud infrastructure are supplied by SaaS [1].

The openness, flexibility, and complex architecture of cloud computing have led to many different types of fault from infrastructure systems, platforms to applications on them. These affect users and cause enormous economic losses. For example, in August 2013 Amazon stopped working in 45 minutes due to a fault which caused losses of up to $5.000.000. According to Tellme Networks, fault detection takes 75% of the system recovery time and prevents 65% of faults from occurring [2]. Thence, the reliability of the service is one of the prerequisite issues when building up cloud computing system. Also, cloud computing services require the ability to identify and to behave appropriately to ensure smoothness and readiness in the face of faults [4], [5]. Having a prior understanding of faults in cloud infrastructure will help minimize the impact of faults on the cloud. Although much research and improvement have been done on cloud computing, some companies have suffered a large amount of downtime as a result of cloud failures resulting in significant revenue loss. According to some researchers, it is likely that in the future, the tool service-level agreement (SLA) Google applications are expected to manage all causes of faults [6]. The system allows operators to set rules for specific parameters to monitor and operate the system monitoring tools of leading companies like Tivoli of IBM, OpenView of HP, CloudWatch of Amazon. The system will then issue alerts when system parameters exceed a set threshold. However, setting thousands of threshold parameters for applications is difficult and depends on the operator's experience. Also, it is hard to have no faults in system because of the large scale of cloud computing data center and its complex architecture. What's more, faults on cloud computing are extremely various with causes to faults existing everywhere in the physical layer. Also, faults have bad effect on abstraction layer. For example, if a fault occurs in the operating system in the PaaS, it may make applications in SaaS fail. Similarly, if faults occur on the IaaS, it will lead to a fault in the operating system of the PaaS. And this will affect faults occurring in the application of the SaaS layer. More unfortunately, faults occur in physical layer like servers, storage and network components, it will gradually affect IaaS, PaaS, and SaaS. It is difficult to collect system parameters from layers such as networks, hardware, operating systems, virtual machines, platforms, ap-

Manuscript received December 23, 2019; revised June 18, 2020; approved for publication by Zhu Han, Division III Editor, July 16, 2020.

This research is funded by Thu Dau Mot University under grant number DT.20-093. Also, we would like to thank Ho Chi Minh City University of Technology (HCMUT) for the support of time and facilities for this study.

K. T. Bui and T. V. Pham are with Faculty of Computer science and Engineering, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Vietnam, email: khietbt@tdmu.edu.vn and ptvu@hcmut.edu.vn.

K. T. Bui and T. V. Pham are with Vietnam National University Ho Chi Minh City, Linh Trung Ward, Thu Duc District, Ho Chi Minh City, Vietnam, email: khietbt@tdmu.edu.vn and ptvu@hcmut.edu.vn.

K. T. Bui and L. V. Vo are with Faculty of Engineering and Techonlogy, Thu Dau Mot University, Binh Duong, Vietnam, email: khietbt, lenvv@tdmu.edu.vn.

C. M. Nguyen is with Key Learning Facilities, Sai Gon University, Ho Chi Minh city, Vietnam, email: canh.nm@lib.sgu.edu.vn.

H. C. Tran is with Training and Science Technology Department, Posts and Telecoms Institute of Technology, Ho Chi Minh city, Vietnam, email: conghung@pithcm.edu.vn.

K. T. Bui is corresponding author.

Digital Object Identifier: 10.1109/JCN.2020.000023

plications from scale of cloud environment with thousands of processing nodes made. In addition, due to the transparency of cloud computing service, it is difficult to analyze the status of the application; because it depends on the interaction of components in the system and its applications. Additionally, fault checking and removing may not totally prevent all faults from occurring [3] since fault detecting and diagnosing face following challenges:

- Cloud computing elasticity and its multi-purposed user services have caused continuous changes in the system models and other apps in cloud computing from time to time, especially in resources allocation.
- Cloud computing transparency makes it difficult to analyze apps status as this much depends on the interaction among the system components together with other apps.
- Cloud computing‘s large scale with thousands of nodes make difficulty in collecting figures of the system from network, hard ware, operating system, virtual machines, infrastructure and apps.
- In practice, collecting abnormal and faults is a problem and it is really costly to make system operate in fault conditions.
- Abnormal and faults in system are tremendous, so it is hardly possible to combine all these types in practice to form training data consisting all abnormal and faults.

Currently, mathematical and statistical models are prominent techniques used for detecting faults. Many different solutions have been suggested to solve this problem such as density-based method [8], [9], neural network based-method, kernel-based method [14] and data mining [7], [10], [11]. Fault detection model is built from monitoring data, history event, and brief system. However, in real applications, collecting all abnormals or faults data is very difficult because of high cost in activity system in fault condition. Therefore, there are very few or no abnormal/fault samples in the training dataset. Also, faults happen in real environment very diversely with species as complication of influencing factors making faults. Prerequisite conditions of technique fault detection must understand procedures and system constructors, especially toward large scale and modern procedures. Most of fault detection methods have focused on modeling normal data in order to figure out faults. During the detecting approach, measures on normal data are done easily. While Schölkopf *et al.* [12] have proposed a method of using a supporting vector in one-class classification named one-class support vector machine (OCSVM), Tax and Duin [13] applied a more effective one called support vector data description (SVDD) in which a hyperplane is replaced with a hypersphere. Due to the ability of generalization of SVDD compared to Neural network, SVM celebrated and was popular in the fields. OCSVM is a particular type of support vector machine (SVM) which needs normal data for training. This is used for forming a decisive boundary with max margin between normal data and the origin. If there is a new object in the boundary, it, then, will be considered a normal datum. Otherwise, it will be grouped into abnormals if it is outside the boundary. OCSVM needs no abnormal data, it is easier to train and to apply to detect abnormal compared to the traditional SVM. However, OCSVM is extremely sensitive to outliers in the training data [15]. Outliers are able to hide abnormals and reduce performance on abnormal detecting. It is hard to clearly identify the boundary with an outlier in the training data. Whenever an outlier occurs, it may be considered as a fault by a popular fault detection method. Thence, labeling input records is used as an outlier detecting method. However, measuring how much outliers influence on input records is not easy as it much depends on data meaning and it leads to the technique used to demonstrate outlier degrees during the process of abnormal detecting. There are some methods to measure the effects of outliers such as Similarity/Distance or relationship model [9], [14]. Yin *et al.* [15] proposed a robust one-class support vector machine for detecting faults based on decisive boundary values when outlier appear in the dataset. The distance between center of the dataset and every sample is calculated to penalty factor of OCSVM model. It is then evaluated to decide whether or not a fault occurring based on a fixed threshold. However, identifying suitable thresholds for decisive boundary values to identify faults is difficult. In addition, setting thresholds leads lack of flexible operation in dynamic cloud environment. To overcome these defects, in this research, the combination of fuzzy logic and OCSVM (named FOCSVM) is proposed to improve the abnormal detection when outliers appear in the dataset. By using fuzzy logic for calculating penalty factors of OCSVM model, fault detection approach improves flexible operations in real time as well as takes advantage of experts' knowledge. Based on the FOCSVM abnormal detection model, the fault detection and diagnosis approach is proposed including abnormal detection, fault detection, and fault diagnosis. For fault detection problem, the exponentially weighted moving average (EWMA) chart is then used to identify abrupt changes if there is any fault to occur like [27], named EWMA-FOCSVM. And then, the fault diagnosis problem is abstracted to feature selection problem with the training dataset which are labeled by EWMA-FOCSVM. The Recursive Feature Elimination (RFE) [42], [43] method which uses Random Forest (RF) algorithm on each iteration is applied to fault diagnosis model, named REF-RF. In experiment, a multi-tier web application based on TPC-W (industry-standard benchmark) which deploys in private cloud evaluate this proposal. Faults are injected the system and are detected by EWMA-FOCSVM. The results of experiment present effectively EWMA-FOCSVM fault detection which benchmark with Threshold-FOCSVM. The RFE-RE model benchmarks to the Bagged Decision Trees algorithm (named RFE-BDT).

The main contributions of the study are as following.

(i) By using fuzzy logic membership function, the adaptive penalties of OCSVM are designed to reduce the effects of outliers. The membership function presents the distance of every normal sample to the center of dataset. This method is called FOCSVM.
(ii) We propose the fault detection approach, named EWMA-FOCSVM, which monitors abrupt fluctuation of the decisive boundary value of FOCSVM based on EWMA chart. This chart is used to detect a small shift in the process in which each point shows a moving average of points.
(iii) The monitored samples are labeled by using EWMA-FOCSVM in real time. Therefore, the training dataset for identifying the metrics causing faults is built, but it is un-

balanced. The fault diagnosis is abstracted to selection feature problem and is solved by using RFE-RF model. The suspicious metrics are ranked this training dataset to find metrics causing faults.

The remainder of this paper is structured as follows. The state-of-the-art is reviewed in Section 2. In Section 3, the system model is represented. The fault detection and diagnosis is formulated in Section 4. Section 5 describes the evaluation. Finally, conclusions are the last section.

II. STATE-OF-THE-ART

A. *Types of Faults in Cloud Computing*

Cloud computing is a distributed system, so its faults are similar to in others. Numerous of faults have been shown in previous research. Jhawar and Piuri [16] have divided faults into two main groups: crash faults and byzantine faults. These faults are described as following:

- Crash faults are those occur by one or many system components like power source, storage disk, chip memory, processor, network switch and router, etc. This type of fault has influence on the physical system and these affected components are manually fixed. Additionally, software agents are deployed for fault tolerance.
- Byzantine faults are caused by differences between the initial expectations and the real results. They, then, directly lead to unpredictable conditions named byzantine. The appearance of byzantine faults affects the logic structure of the system and may not require manual interaction as they may be processed by proper fault tolerance strategies. It may be hard to check and conclude exactly on byzantine conditions due to the system's complexity and variety. It is hardest to detect a system providing incorrect result to customers. The replicas of system are deployed to avoid this faults.

Numerous types of faults are identified in the document dedicated to cloud environments [17], [20], [27] which can be classified into: data faults, calculation faults and response time faults:

- Data faults are faults due to data collection during the operation of system. The data may be jammed, incomplete, and even some damaged data may be unreadable. For example, the online customer information can be wrong because hackers falsify information like a "robot" generates thousands of fake information.
- Calculate faults occur in the network due to network partition, packet loss, packet failure, destination fault, link fault,etc; faults occur in hardware such as CPU faults, faults in memory, etc; faults occur in the processor, operating system, calculation software,etc; process fault due to lack of resources.
- Response time faults do not show results like an error page is displayed rather than the log-in page or a message "Enter the wrong password" when we log on to a website to shop. This fault may occur due to data faults or calculation faults.

B. *Fault Dtection and Diagnosis in Cloud Computing*

Fault detection problem is one of the biggest challenges of a real system. It is an important task to ensure reliability and to reduce losses which happen by fault in the system. System fault detecting and diagnosing has attracted lots of attention from many scientists as well as businessmen and investors. Millions of dollars has been spent on it with many worldwide big projects conducted by professors, engineers and workers. This proves how much important of system fault detecting and diagnosing is in both research and economics as well. Fault detection in cloud computing cannot be found manually which is performed by a system. There are two main strategies including intrusion detection and Heartbeat/Ping to build fault detection system.

- Intrusion strategy: The variables and status of the system are observed, collected, and measured periodically. After that, the process of analyzing and evaluating the variables and the state of the system are applied to quantify information. Intrusions are identified by measuring a bias between the measured values of system with those measured in the normal process. There are statistical data mining and machine learning techniques to monitor this bias. The abnormal behaviors are detected by using the model of normal behaviors of users in which are deviated from. The objects monitored and analyzed for detecting intrusion are network packet attributes (e.g., IP address, network volume and processing status) associated with the data from the features of the web application (e.g., request chain, transaction completion and request processing).
- Heartbeat strategy: A message is sent periodically from a monitored node to the bug detector to notify that it is still alive. If Heartbeat' messages do not arrive before the time was up, the fault detector suspected that there is an occurred fault at this node.
- Pinging strategy: A message is sent continuously from a fault detector to a monitored node. The fault detector will get the answer ACK (Acknowledgment). If a sustained message fails, a probe (a series of messages separated by a period of time) can be used to verify if a node is actually faulty. Detecting broken nodes with partially centralized monitoring supports to large-scale system. However, inability to detect malicious attacks if the nodes are still functioning properly, detecting incidents depending on the reliability of the detection node and network conditions, potentially buffer overflow due to transmission the message.

Most of the current fault detection techniques are based on system monitoring and known as fault samples [21]–[23], [25], [26]. Known faults are defined and turned into specific set of rules. Fault diagnosing will later be operated by comparing signatures to the existing rules. System's signatures are collected by system operators or commercial monitoring tools like HP OpenView, IBM Tioli, etc., so that alerts will automatically generate whenever metric values exceed the predefined thresholds. Web applications provide concurrent services to a large number of users through Internet connections. Users make interactions, if at the same time the number of inter-acting users soars, the workload of the web application will increase dramatically leading to anomalies in the workload of the web application. Wang *et al.* in [26] proposed an algorithm which online workload samples were trained and used local exceptions in the workload pattern detect abnormal workload of web application. Lin *et al.* [21] have suggested a mechanism on IaaS cloud computing fault detection by extracting data performance using global locality preserving projection. Local outlier factor is then applied to detect abnormals. OpenStack and Xen are applied to conduct a cloud computing platform system for evaluating fault injection. System's real time data performance was collected and analyzed continuously. And then, F-measure was operated to measure the accuracy of the algorithm. Lin *et al.* have experimented the efficient data-driven and local outliers factor-LOF in order to recognize abnormal performance as well as figure out metrics causing performance abnormal during the experiment. Experiments indicated that global locality preserving projection works more effectively than PCA algorithm and others in terms of precision, recall and F-measure. Mounya Smara *et al.* proposed a fault detection framework for component-based cloud computing using acceptance test [23]. This framework detects faults like transient hardware faults, software faults, and response time faults made locally on each computer in the cloud system. In the acceptance testing strategy, each cloud node has its own acceptance check that can be considered an internal monitoring of node behavior; if any abnormal behavior is detected at the node, it will be considered intrusive and blocked immediately. Pinto *et al.* in [22] proposed the fault prediction by using SVM model. SVM model which analyze labeled dataset is supervised learning model for classification. System faults are predicted earlier by SVM model which monitors metrics on servers. Therefore, jobs are done earlier by rescheduling based on fault prediction. In addition, a reinforcement learning module is designed to avoid false positives. Zhang *et al.* [25] proposed an online fault detection approach that depends on SVM-Grid to predict emerging issues in cloud. The accuracy of traditional SVM model is enhanced by using grid method for model's input parameters to accomplish fine-tuned prediction. In addition, FT algorithm which updates samples database to minimize time costs is developed. To evaluate the proposed method, Ft algorithm is benchmarked with Back Propagation, traditional SVM, and Learning vector quantization (LVQ). FT algorithm has shown to have the highest accuracy one.

However, in large-scale system with thousands of metrics, it is difficult to define faults as well as find suitable thresholds. There are some methods to solve this problem which do not know about faults previously [24], [27]. Sometimes, the system is operated normally, suddenly warning faults or fault occurs [24]. Apart from classical or known faults, it is difficult to determine what is happening. A research used fault injection and assumed that this injection fault will correspond to the actual error of the system. So, it is entirely possible to rely on the information from this fault to determine the root caused for the fault to be alerted in the actual system. This method is implemented by combining fault analysis and data analysis, and it has ability to find previously unknown faults. Another work proposed the method of fault recognition based on the canonical correlation analysis (CCA) [27]. This research proposed an online incremental clustering method to identify access behavior patterns and used CCA to model the correlation between workload and performance metrics/resources in a specific access behavior pattern. From this correlation, a coefficient is found that the difference between two consecutive coefficients will tell us that an error exists there. In [19] proposed the online abnormal detection method based on self-adaptive cloud monitoring. To address the issues which administrators manually define suitable monitoring rules, the self-adaptive monitoring approach is built with two phases. First, the correlations between metrics is calculated to reflect the running status of a system. Second, the running status is characterized by using principal component analysis (PCA) to estimate the anomaly degree, and to predict the possibility of faults. By adjusting automatically in the monitoring period, the accuracy of abnormal detection is improved. This approach has low overhead. Differently, we proposed the method which detects faults by monitoring the decisive boundary values of Fuzzy One-Class support vector machine model.

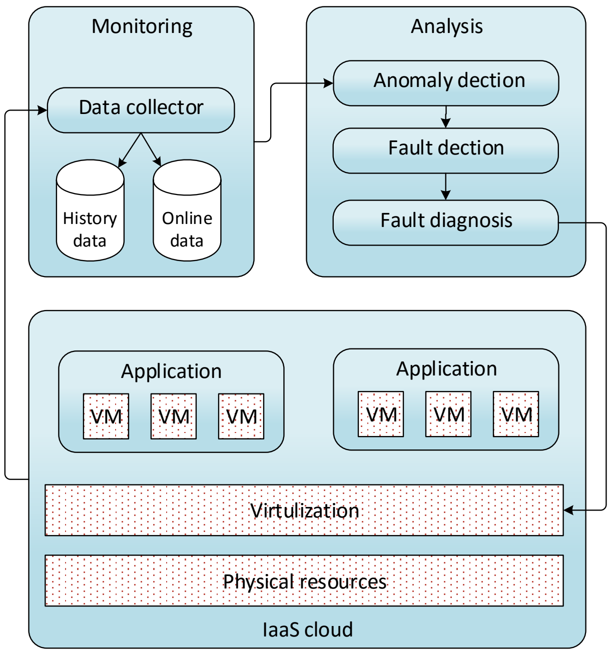

Fig. 1. System architecture.

III. FAULT DETECTION AND DIAGNOSIS APPROACH

A. *System Architecture*

Web applications are hosted on cloud computing which usually possess multi-tier architecture such as Servlet, JavaServer Pages and Enterprise JavaBeans based on Java Framework or .Net Framework of Microsoft. A typical multi-tier architecture usually consists of a view tier, business logic tier and data base tier. There are specific servers for each layer which is a virtual machine (VM) for applications in cloud computing infrastructure services. Fig. 1 shows the system architecture which includes three main components Infrastructure as a Service (IaaS) cloud, monitoring and fault analysis.

A.1 Monitoring Component

Monitoring component collects data from physical resources and virtualization. Metrics of system performance such as resource usage of servers, workloads, and performance metrics in the application are gathered by interfaces provided by operation system or third-party software like Hyperic SIGAR, Ganglia, or Promethus etc. The collected data will be first processed by standardizing, outliers adjusting and deleting those repeated. Next, the gathered data is both stored in historical and online ones. Finally, online data will be transformed into Fault analysis component while the historical one will be labeled as normals or faults to be used for fault diagnosis. After that, the monitoring data will be saved to the archive (historical data).

A.2 Fault Analysis Component

There are three main phases in fault analysis component including abnormal detection, fault detection and fault diagnosis.

- Abnormal detection is a process to identify objects with bias compared to common ones. These objects carry significant differences or are created by a different mechanism compared to the normal ones. It is essential to find out a proper method of deviation measurement serving for abnormal detection purpose.
- Fault detect is a component to monitor the characteristic system state and then identify abnormal data related to faults by comparing the multidimensional monitoring data collected. Sudden changes in the metrics are detected using EWMA control charts that do not require specialized knowledge.
- Fault diagnosis is a component which applies feature selection method through analyzing variance of multiple parameters. And then, auto-scaling technologies (e.g., scale-up, scale-down, migration) is used to reduce the impact of faults through adjusting allocation resources in the IaaS cloud.

B. Abnormal Detection Based on Fuzzy One-Class Support Vector Machine

In this section, the fuzzy one-class support vector machine abnormal detection model is proposed based on the general one-class support vector machine first proposed by Schölkopf *et al.* In many real-world applications, types of training dataset have different effects on training models like some training points which are more important than others. In particular, the target datas*et al.*ways includes some outliers for the reasons of non-representative sampling or instrument failures. In one-class SVM, the slack variables $\xi = [\xi_1, \xi_2, \cdots, \xi_N]$ are used to locate some data points outside the decisive boundary. The number of points located outside the decisive boundary can be controlled by the penalty factors of $1/Nv$. However, the normal data points and outliers which have the same penalty factors in the optimization problem in (1) lead to the shifting toward the decisive boundary in the outliers trend. Tuning down the penalty factors reduces the effects of outliers. Consequently, the classification accuracy of one-class SVM is badly affected by a part of normal data points outside the decisive boundary. In order to improve the one-class SVM with outliers, the adaptive penalty factors are designed based on the fuzzy membership function of a data point with outliers occurred. The adaptive penalty factors based on fuzzy membership function describes the distances of relationship between a data point and the center of the training dataset. Calculating the center of the training data is applied the total square loss center (tSL-center) model proposed by Liu *et al.* [30]. In [31], the experiments presented the ability of tSL-center to outliers. To solve Fuzzy one-class support vector machine problem, the SMO algorithm is applied to select two α parameters, α_a and α_b and to optimize the objective value [32]. The output of abnormal detection phase is used to identify faults in the fault detection phase.

B.1 One-class Support Vector Machine

Schölkopf *et al.* proposed a method for forming a decisive boundary which has the maximum margin between the origin and the normal data set. Considering the dataset of $X = [x_1, x_2, \cdots, x_N] \in \Re^{N_x M}$, which accommodates normal data, an optimization model of the boundary is considered as follows:

$$\begin{gathered}\min_{w,\xi,\rho} \frac{1}{2}\|w\|^2 + \frac{1}{Nv}\sum_{i=1}^{N}\xi_i - \rho \\ subject\ to\ \ w.\Phi(x_i) \geq \rho - \xi_i,\ \ \xi_i \geq 0,\end{gathered} \tag{1}$$

where N is the number of the instances, $v \in (0, 1]$ is a regularization parameter and $\xi = [\xi_1, \xi_2, \cdots, \xi_N]$ is nonzero slack variables penalized in the objection function. ρ and w are the parameters which determine the decisive boundary and they are target variables of the objection function. The decisive boundary can be formulated as:

$$f(x) = sgn[w.\Phi(x) - \rho]. \tag{2}$$

The dataset of the classification problem is not always linearly separable in original space. However, with the Φ, the original dataset can be linearly separable in high dimensional space. However, it is difficult to find Φ in the practical application and k kernel function representing the dot product form k(x,y) is necessary to be known,

$$k(x, y) = \Phi(x).\Phi(y). \tag{3}$$

The commonly used kernel functions follow as:
- Linear kernel: $k(x, y) = x^T y$
- Polynomial kernel: $k(x, y) = (x^T - c)^d$
- RBF kernel: $k(x, y) = exp(-\frac{\|x - y\|^2}{2\sigma^2})$

where c is constant, d is the degree of polynomial and σ is the width of radial function (RBF) kernel.

Lagrangian theorem is used to solve the problem of optimization in (1), and the dual problem becomes:

$$\begin{gathered}\min \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j k(x, y) \\ subject\ to\ \ \sum_{i=1}^{N}\alpha_i = 1,\ \ 0 \leq \alpha_i \leq \frac{1}{Nv},\end{gathered} \tag{4}$$

where $\alpha_i \geq 0 (i = 1, \cdots, N)$ is multiplier Lagrangian

Together with (3), the $f(x)$ function (2) becomes:

$$f(x) = sgn\left[\sum_{i=1}^{N}\alpha_i k(x_i, x) - \rho\right]. \tag{5}$$

The $f(x)$ function takes the value $+1$ in a region capturing most of the data points and -1 elsewhere [28]. The α is obtained by solving the optimization problem (4) and then ρ can be given as:

$$\rho = \frac{1}{n_s}\sum_{i=1}^{N_s}\sum_{j=1}^{N}\alpha_j k(x_j, x_i), \tag{6}$$

where n_s is the number of support vectors which satisfy $\xi_i = 0 \ and \ 0 < \alpha_i < \frac{1}{Nv}$.

B.2 Fuzzy One-class Support Vector Machine

Supposing a set of training points with associated fuzzy membership are given

$$X = (x_1, \lambda_1), \cdots, (x_N, \lambda_N). \tag{7}$$

The optimization model of fuzzy one-class SVM is then formulated as

$$\begin{aligned} &\min_{w,\xi,\rho} \frac{1}{2}\|w\|^2 + \frac{1}{Nv}\sum_{i=1}^{N}\lambda_i\xi_i - \rho \\ &subject\ to\ \ w.\Phi(x_i) \geq \rho - \xi_i, \ \ \xi_i \geq 0, \end{aligned} \tag{8}$$

in here $0 < \lambda_i \leq 1$ is a fuzzy membership to shows the level of outlier influence for each training point x_i, the slack variables ξ_i describes a measure of error in OCSVM, and the term $\lambda_i\xi_i$ describes a measure of error with different weighting [29].

The adaptive penalty factors related to the distances of relationship between x_i and the center of the training dataset. The distances can be calculated as

$$d_i = \|x_i - \mathbf{C}\|^{\mathbf{2}}, \tag{9}$$

in which $\mathbf{C}$ is the center of the training dataset. The $\mathbf{C}$ can be expressed as

$$\mathbf{C} = \sum_{i=1}^{N} k_i x_i, \tag{10}$$

where

$$k_i = \frac{\frac{1}{\sqrt{1+4\|x_i\|^2}}}{\sum_{j=1}^{N}\frac{1}{\sqrt{1+4\|x_j\|^2}}}. \tag{11}$$

Substituting (10) into (9), we have

$$d_j = \|x_i\|^2 - 2x_i\sum_{j=1}^{N}k_j x_j + k^T Hk. \tag{12}$$

Using $k(x_i, x_j)$ to replace $x_i.x_j$ into (12)

$$d_j = k(x_i, x_j) - 2\sum_{j=1}^{N}k_j k(x_i, x_j) + k^T Hk, \tag{13}$$

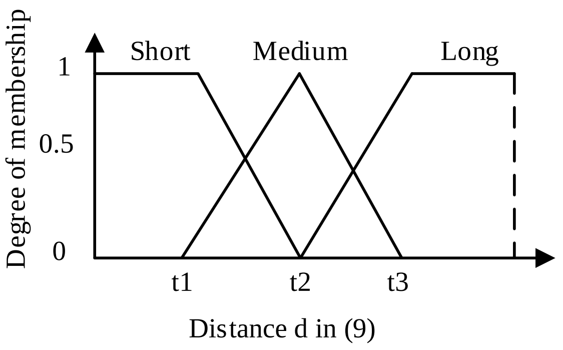

Fig. 2. Membership function of distance.

where $k = [k_1, k_2, \cdots, k_N]$ with k_i is

$$k_i = \frac{\frac{1}{\sqrt{1+4k(x_i, x_i)}}}{\sum_{j=1}^{N}\frac{1}{\sqrt{1+4k(x_j, x_j)}}}. \tag{14}$$

In this study, we propose the adaptive penalty factors based on fuzzy membership function to the distances of relationship between x_i and the center of the training dataset. Outliers have smaller penalty factors λ_i than the normal data points if they are far from the center. The distances are fuzzified by the triangular membership function associated with three kinds of fuzzy sets $F_k, k = 1, 2, 3$ in which variable languages of Short, Medium and Long are measured as illustrated in Fig. 2.

$$\begin{aligned} MF_1(x_i) &= \max\left\{\frac{x_i - a_2}{a_1 - a_2}, 0\right\} \\ MF_2(x_i) &= \min\left\{\frac{x_i - a_2}{a_1 - a_2}, \frac{x_i - a_2}{a_2 - a_3}\right\} + 1 \\ MF_3 &= \max\left\{-\frac{x_i - a_2}{a_2 - a_3}, 0\right\} \\ a_1 = \max_{i=1,\cdots,N} x_i, a_3 &= \min_{i=1,\cdots,N} x_i, a_2 = (a_1 + a_3)/2, \end{aligned} \tag{15}$$

the membership function of the distances are calculated as followed

$$g_k(d_i) = \frac{MF_k(d_i)}{\sum_{k=1,2,3} MF_k(d_i)}, \ (i = 1, \cdots, N). \tag{16}$$

In order to solve the optimization problem (8), Lagrange multipliers $\alpha_i \geq 0, \gamma_i \geq 0 (i = 1, 2, \cdots, N)$ are introduced and the Lagrange equation is formed as:

$$\begin{aligned} L(w, \rho, \xi, \alpha, \gamma) &= \frac{1}{2}\|w\|^2 + \frac{1}{Nv}\sum_{i=1}^{N}(\lambda_i\xi_i - \rho) \\ &- \sum_{i=1}^{N}\alpha_i[(w.\Phi(x_i)) - \rho + \xi_i] - \sum_{i=1}^{N}\gamma_i\xi_i. \end{aligned} \tag{17}$$

The partial derivatives of the Lagrangian equation with respect to w, ρ and ξ are set to zero:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N} \alpha_i \Phi(x_i) = 0, \quad \Rightarrow w = \sum_{i=1}^{N} \alpha_i \Phi(x_i) \tag{18}$$

$$\frac{\partial L}{\partial \rho} = 1 - \sum_{i=1}^{N} \alpha_i = 0 \quad \Rightarrow \sum_{i=1}^{N} \alpha_i = 0, \tag{19}$$

$$\frac{\partial L}{\partial \xi} = \lambda_i \frac{1}{N\upsilon} - \alpha_i - \gamma_i = 0 \quad \Rightarrow \alpha_i = \lambda_i \frac{1}{N\upsilon} - \gamma_i \ \ and \ \ \alpha_i \le \lambda_i \frac{1}{N\upsilon}. \tag{20}$$

Substituting (18) (20) into (17) and its dual form is presented as:

$$\min_{\alpha} \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j k(x_i, x_j) \quad subject\ to \ \sum_{i=1}^{N} \alpha_i = 1, \quad 0 \le \alpha_i \le \lambda_i \frac{1}{N\upsilon}, \tag{21}$$

B.3 SMO Algorithm for Fuzzy One-class Support Vector Machine

By breaking a large quadratic problem into a series of size two quadratic problems, the objective function of (21) can be rewritten as:

$$\begin{aligned} L(\alpha_a, \alpha_b) =& \frac{1}{2}{\alpha_a}^2 k(x_a, x_a) + \frac{1}{2}{\alpha_b}^2 k(x_b, x_b) \\ &+ \alpha_a \alpha_b k(x_a, x_b) \\ &+ \sum_{i=a,b} \alpha_i \sum_{j=1, j\neq a,b}^{N} \alpha_b k(x_i, x_j) + L', \end{aligned} \tag{22}$$

where L' is a term of strictly constant with respect to α_a, α_b. Due to the equality constraints of optimization problem, we let

$$s^* = \alpha_a^* + \alpha_b^* = \alpha_a + \alpha_b \tag{23}$$

We get

$$\alpha_a = s^* - \alpha_b. \tag{24}$$

Substituting (24) into (22)

$$\begin{aligned} L(\alpha_b) =& \frac{1}{2}(s^* - \alpha_b)^2 k(x_a, x_a) + \frac{1}{2}\alpha_b^2 k(x_b, x_b) \\ &+(s^* - \alpha_b)\alpha_b k(x_a, x_b) + \alpha_b \sum_{j=1, j\neq a,b}^{N} \alpha_j k(x_b . x_j) \\ &+(s^* - \alpha_b) \sum_{j=1, j\neq a,b}^{N} \alpha_j k(x_a . x_j) + L'. \end{aligned} \tag{25}$$

The partial derivatives of the Lagrangian equation with respect to α_b

$$\begin{aligned} \frac{\partial L(\alpha_b)}{\partial \alpha_b} =& (\alpha_b - s^*)k(x_a, x_a) + \alpha_b(x_b, x_b) \\ &+(s^* - 2\alpha_j)k(x_a, x_b) + \sum_{j=1, j\neq a,b}^{N} \alpha_j k(x_b, x_j) \\ &- \sum_{j=1, j\neq a,b}^{N} \alpha_j k(x_a, x_j). \end{aligned} \tag{26}$$

Let $\dfrac{\partial L(\alpha_b)}{\partial \alpha_b} = 0$, we get

$$\begin{aligned} \alpha_b =& \eta[s^*(k(x_a, x_a) - k(x_b, x_b)) \\ &- \sum_{j=1, j\neq a,b}^{N} \alpha_j k(x_b, x_j) \\ &- \sum_{j=1, j\neq a,b}^{N} \alpha_j k(x_a, x_j)], \end{aligned} \tag{27}$$

where

$$\eta = \frac{1}{k(x_a, x_a) + k(x_b, x_b) - 2k(x_a, x_b)}. \tag{28}$$

Let

$$\bar{f} = \sum_{i=1}^{N} \alpha_i k(x_i, x) - \rho, \tag{29}$$

we get

$$\begin{aligned} \alpha_b =& \eta[s^*(k(x_a, x_a) - k(x_b, x_b)) - \bar{f}^*(b) + \bar{f}^*(a) \\ &+ (s^* - \alpha_b)k(x_b, x_a) + \alpha_b^* k(x_b, x_b) \\ &- (s^* - \alpha_b^*)k(x_a, x_a) - \alpha_b^* k(x_a, x_b)] \\ =& \eta[\bar{f}(a) - \bar{f}(b) + \alpha_b^*(k(x_b, x_b) + k(x_a, x_a) - 2\alpha_b^*] \\ =& \alpha_b^* + \eta[\bar{f}^*(a) - \bar{f}^*(b)]. \end{aligned} \tag{30}$$

The (30) is the rule for updating α_b.

To ensure upper bound and lower bound of α_a, α_b with $0 \le \alpha_a \le \lambda_i \frac{1}{N\upsilon}$, $0 \le \alpha_b \le \lambda_i \frac{1}{N\upsilon}$, we use:

- $L = \max(s^* - \lambda_i \frac{1}{N\upsilon}, 0)$
- $H = \min(\lambda_i \frac{1}{N\upsilon}, S^*)$

Karush-Kuhn-Tucker (KKT) condition is used to check the optimization of SMO. Before each iteration, KKT is used to check whether α_i needs updating or not including:

- $\alpha_i(\omega.\Phi(x_i)) - \rho + \xi_i = 0$
- $\gamma_i \xi_i = 0$

Regarding an optimal solution, KKT includes three cases:

* **Case 1** $\alpha_i = 0$

We have $\gamma_i = \lambda_i \frac{1}{N\upsilon} - \alpha_i \ \rightarrow \ \gamma_i \neq 0 \ \rightarrow \ \xi_i = 0$

Therefore, $\alpha_i(\omega.\Phi(x_i)) - \rho + \xi_i > 0$ then $\alpha_i(\omega.\Phi(x_i)) - \rho > 0$

$\rightarrow \bar{f}(x_i) > 0$

* **Case 2** $0 < \alpha_i < \lambda_i \frac{1}{N\upsilon}$

We have $\gamma_i = \lambda_i \frac{1}{N\upsilon} - \alpha_i \rightarrow \gamma_i \neq 0 \rightarrow \xi_i = 0$

Therefore, $\alpha_i(\omega.\Phi(x_i)) - \rho + \xi_i > 0$ then $\alpha_i(\omega.\Phi(x_i)) - \rho = 0$
$\rightarrow \bar{f}(x_i) = 0$

* **Case 3** $\alpha_i = \lambda_i \frac{1}{N\upsilon}$

We have $\gamma_i = \lambda_i \frac{1}{N\upsilon} - \alpha_i \rightarrow \gamma_i = 0 \rightarrow \xi_i \neq 0$

Therefore, $\alpha_i(\omega.\Phi(x_i)) - \rho + \xi_i > 0$ then $\alpha_i(\omega.\Phi(x_i)) - \rho < 0$
$\rightarrow \bar{f}(x_i) < 0$

From the three cases above, a α_i is optimized if one of the following three conditions is fulfilled.

- $\alpha_i = 0 \wedge \bar{f}(x_i) > 0$
- $0 < \alpha_i < \lambda_i \frac{1}{N\upsilon} \wedge \bar{f}(x_i) = 0$
- $\alpha_i = \lambda_i \frac{1}{N\upsilon} \wedge \bar{f}(x_i) < 0$

We consider the equality constraint condition of (21), then the Lagrange function can be performed as following:

$$\bar{L} = \sum_{i,j=1}^{N} \alpha_i \alpha_j k(x_i, x_j) + \delta(1 - \sum_{i=1}^{N} \alpha_i). \tag{31}$$

The Lagrange multiplier $\delta = \rho$ with ρ in (29), so the (31) can be presented as following:

$$\bar{L} = \sum_{i,j=1}^{N} \alpha_i \alpha_j k(x_i, x_j) + \rho(1 - \sum_{i=1}^{N} \alpha_i). \tag{32}$$

In the (32), gradient of objective function with α_b is demonstrated as

$$\frac{\partial \bar{L}}{\partial \alpha_b} = \sum_{i=1}^{N} \alpha_i k(x_i, x_b) - \rho = \bar{f}(x_b). \tag{33}$$

For the target function, the greater absolute value of the gradient is, the larger the function oscillation becomes when the optimal variables change.The first optimal variable is chosen based on this criterion. For others, the criterion of how to make the oscillation variables greatest which is suggested by Scholkopf is taken into consideration. Therefore, the two-step-heuristic strategy for selecting value pairs is calculated as following:

- Firstly,the first variable of α_b is chosen by browsing through the entire training and being checked to see if any of the samples violate KKT and create $Max(|\bar{f}(x_b)|)$.
- Secondly, the second variable of α_a is continually counted by maximizing the steps performed during the optimization process of $Max(|\bar{f}(x_b) - \bar{f}(x_a)|)$.

C. Fault Detection with EWMA Chart

In this section, the fault detection model, named EWMA-FOCSVM, is modeled by using the output of the abnormal detection model and EWMA chart. The decision value of $f(x)$ in (5) will be stable in a normal situation. Otherwise, it will fluctuate much when faults are triggered. In dynamic cloud environment, the fluctuation of decision values need to be figured out if they are stable or not. To do this, these values are monitored, and then an alarm is set when the current values are inconsistent with the past ones. For example, Web applications run normally in most cases, but they become abnormal when there is a huge of sudden users' requests at the same time or the systems are attacked by "hackers". Thus, a control chart is needed to track and calculate the stability of the decision values. In statistical quality control, the EWMA chart is used to monitor variables using process's entire output [38]. Each data point in the EWMA represents a moving average of points. This chart is appropriate for detecting a small shift in the process. In addition, EWMA has lower computational cost than the abrupt change detection technologies [27]. Furthermore, EWMA requires no knowledge for fault detection based on thresholds. The output of fault detec-

Algorithm 1 Abnormal detection - Training model with SMO

Input:
Data instances: $X = [x_1, \cdots, x_n]^T$
Output:
The abnormal detection model

```
1:  Initialization $\alpha_a^{old}$, $\alpha_b^{old}$
2:  while $\neg KKTCondition$ do
3:      Choose two parameter $\alpha_a$,$\alpha_b$
4:      Calculate $\alpha_b^{new}$ by equation (30)
5:      if $\alpha_b^{new} > H$ then
6:          $\alpha_b^{new} = H$
7:      end if
8:      if $\alpha_b^{new} < L$ then
9:          $\alpha_b^{new} = L$
10:     end if
11:     Calculated $\alpha_a^{new} = \alpha_a^{old} + \alpha_b^{old} - \alpha_b^{new}$
12:     Calculated $\rho$ follow equation (6)
13: end while
```

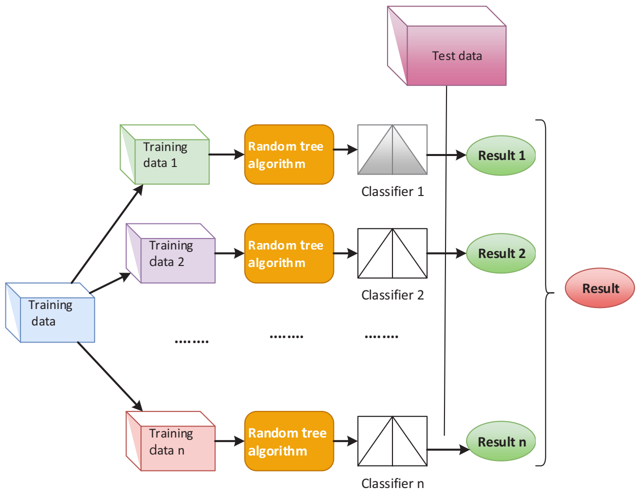

Fig. 3. Random Forest feature selection model.

tion model helps to label training data in fault diagnosis phase.

The decisive values $f(x)$ in 5 are measured with EWMA in fault detection. EWMA formula is as following:

$$Z_i = \tau f_i + (1 - \tau) Z_{i-1}. \tag{34}$$

where $0 < \tau < 1$ is the smoothing constant, and Z_i is the EWMA statistic for the ith, f_i is the ith decisive value, and Z_0 is the average value of the initial decisive values:

$$\begin{aligned} UCL_x(i) &= \mu_z + L_z \sigma_z \\ LCL_x(i) &= \mu_z - L_z \sigma_z, \end{aligned} \tag{35}$$

where UCL is the upper control limit, LCL is the lower control limit. When $Z_i < LCL_{x(i)} or Z_i > UCL_{x(i)}$, an alarm of fault will be set in the system.

D. Fault Diagnosis with Feature Selection Method

In this section, the fault diagnosis model named RFE-RF is presented by adopting a feature selection method. Fault roots are automatically located by analyzing the online data monitored before and after the faults occurred. Faults play key roles in increasing the metrics fluctuation, so identifying fluctuation metrics is vital to find the fault roots. There are many types of metrics in a monitored sample. However, the metrics causing faults contain high fluctuations are those need fixing in fault tolerance. And then, the result of high fluctuation metrics positioning will be abstracted to the feature selection problem. The training dataset for the fault diagnosis problem is built based on the output of fault detection phase. After detecting faults, the online monitoring data is labeled as fault and as normal before faults occur. However, the training dataset of this problem is usually unbalanced because there are more normal samples than the faulty ones. In unbalanced training dataset cases, the RF algorithm was proved suitability [33], [39]. RF is an extended of decision tree which classifies data by building a number of classifiers (decision trees) to reach high accuracy. The classifiers are constructed for a given task, and then they are combined to a new classifier. However, to improve the performance of RF algorithm, the RFE method [42], [43] which uses RF algorithm on each iteration is applied to rank the suspicious metrics related to faults.

The fault diagnosis problem is shown as follows:

$$\mathcal{L} = \mathcal{F}(Normal, Fault) \tag{36}$$

where $\mathcal{L}$ is set of suspicious metrics, $Fault$ is fault dataset, $Normal$ is the normal dataset and $\mathcal{F}$ is a feature selection method. As shown in Table 1, the samples are labeled $Normal$ before fault detecting at 14:43:30, and the samples after detecting named $Fault$.

As shown in Fig. 3, training data are sampled randomly into n groups from Training data $1, 2, \cdots, n$ in order to form decision trees of Classifier $1, 2, \cdots, n$ by using the decision tree algorithm. These decision trees are then combined into random forest models which are applied to classify data into specific purposes named final result. Random training data is one way of reducing the extent of this imbalance [40].

The fault diagnosis algorithm is showed in Alg.2 in which the features are ranked by RF algorithm. Given $L = \{(X_i, Y_i)_{i=1}^N | X_i \in R^M, Y_i \in \{Normal, Fault\}\}$, where X_i are predictor variables, Y_i is a class of feature, N is the number of instances in training dataset, M is the number of features. Each tree is built from a bagged sample set which is two-third of the samples in L (called in-bag sample). The rest of the samples in L, called out-of-bag (OOB), are used to estimate the prediction error. The prediction of random forest with K is:

$$\hat{Y} = \mathcal{V}\{\hat{Y}^k\}_1^K \tag{37}$$

where $\hat{Y}^k$ is the prediction of k_{th} tree, and $\mathcal{V}$ is voting method.

E. Fault Detection and Diagnosis Scheme

In this section, the fault detection and diagnosis scheme including training phase and online processing phase is proposed by using the abnormal detection model, fault detection model, and fault diagnosis model. In the training phase, there are four steps including data normalizing, membership function calculating, abnormal detection model solving, and ρ parameter calculating. There are five steps in the online processing phase when a new sample arrives. These steps include data normalizing, the boundary decision calculating, fault detection based on EWMA-FOCSVM model, locating fault roots when faults occur, and training dataset will be updated in the next time.

E.1 Training Phase

Given a training data set X, the model of fuzzy one-class SVM can be formulated as:

1) The columns $X = [x_1, \cdots, x_N]^T \in R^{(NxM)}$ are normalized to zero mean and unit variance. By using PCA, the normalized data set obtain its score vector T:

$$T = XP_{pc} = [t_1, \cdots, t_N]^T \in R^{Nxl}, \tag{38}$$

where $P_{pc} = [p_1, \ldots, p_l] \in R^{(Nxl)}$ represents the principal component matrix.

2) Calculate the crisp out of the data points based on membership function fuzzy logic $g_k(d_i)$

3) Train FOCSVM model based on T

$$\begin{aligned} &\min_{\alpha} \alpha^T H \alpha \\ &\textit{subject to} \ \sum_{i=1}^{N} \alpha_i = 1, \quad 0 \le \alpha_i \le y_i \frac{1}{N\upsilon}, \end{aligned} \tag{39}$$

where

$$H_{ij} = k(t_i, t_j) = \Phi(t_i).\Phi(t_j). \tag{40}$$

4) Calculate the constant ρ as

$$\rho = \frac{1}{n_s} \sum_{i=1}^{n_s} \sum_{j=1}^{n} \alpha_j k(t_i, t_j), \tag{41}$$

where t_i are the support vectors with $\xi_i = 0$, n_s is the number of support vectors, t_j denote the score vectors.

Table 1. Dataset for feature selection problem.

Time	CPU idle (%)	Memory used (MiB)	Page in (kBs)	Inbound (kBs)	...	Label
...	...	...	...	...	...	...
14:42:30	95.85	548	343	370	...	Normal
14:42:45	97.5	551	352	383	...	Normal
14:43:00	97	549	348	371	...	Normal
14:43:15	80.8	548	360	372	...	Normal
14:43:30	**82**	**551**	**398**	**360**	**...**	**Fault**
14:43:45	**88**	**549**	**380**	**332**	**...**	**Fault**
...	...	...	...	...	...	...

Algorithm 2 Fault diagnosis - Feature rank with Random Forest

```
Input:
Data instances: $X = [x_1, \cdots, x_n]^T$
Label instances: $Y = [y_1, \cdots, y_n]^T$, $y_i \in \{Normal, Fault\}$
Training dataset: $L = \{(X_i, Y_i)_{i=1}^N\}$
The number of trees: $K$
The size of the subspace: $S$
Output:
Feature ranked list
1:  for $k \leftarrow 1$ to $K$ do
2:      Draw a bagged subset of sample $L_k$ from $L$
3:      while stopping criteria is not met do
4:          Select randomly $S$ features
5:          for $s \leftarrow 1$ to $S$ do
6:              Compute the decrease in the node impurity
7:          end for
8:          Choose the feature which decreases the impurity most
            and the node is divided into children nodes
9:      end while
10: end for
11: Combine the $K$ trees to a random forest model
```

E.2 Online Processing Phase

The online computation for fault detection when a new sample arrives, i.e. $x \in R^{1xM}$, consists of:
1) The score vector t is obtained by normalizing the new sample with the mean and variance of X

$$t = xP_{pc} \in R_{1xl}. \tag{42}$$

2) Calculate the boundary decision function

$$F(t) = -\sum_{i=1}^{n_s} \alpha_i k(t_i, t) + \rho, \tag{43}$$

where t_i are the support vectors with $xi_i = 0$, n_s is the number of support vectors.
3) Using EWMA-FOCSVM model to detect a fault to $F(t)$
4) Using RFE-RF model to ranking suspicious metrics after detecting fault.
5) Updating training dataset after detecting fault.

IV. EVALUATIONS

A. Environment

Table 2. Confusion matrix.

Actual/Predict	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

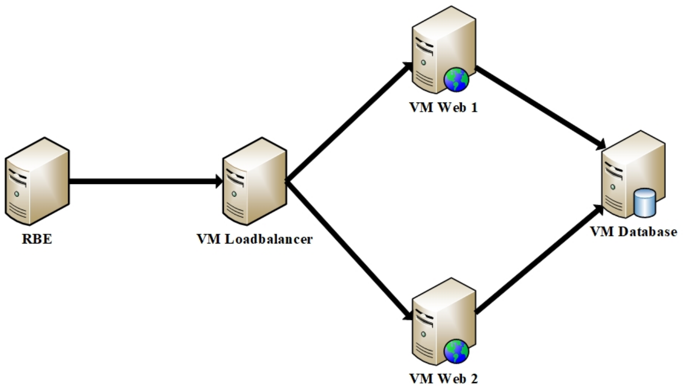

Fig. 4. TPC-W e-commerce application system.

To evaluate methods of detecting and identifying faults, TPC-W[1] - open source e-commerce application on service of cloud computing- was implemented. TPC-W is a benchmark for E-commerce web, including three main components: (i) Web Application System Under Test (STU); (ii) workload generator and send to STU; and (iii) performance monitoring component for SUT. TPC-W allows simulation of three different types of web interactions: shopping (WIPS), web browsing (WIPSb) and web-based ordering (WIPSo). TPC-W uses the concept of an Emulated Browsers (EB) group to make requests to the SUT. EB simulates users interacting with SUT using a browser by sending and receiving HTML content through HTTP. The number of EB used for a test is determined by the size and ratio factor of the SUT, which is constant throughout the test. The workload generated by EB is indicated by the navigation patterns in a session and its workload intensity. TPC-W defines user session duration as the time elapsed between the first transaction made by an EB and the current time.

Fig. 4 demonstrates four VMs which run Ubuntu 18.04 with 1 VCPU and 2 GB RAM, 40GB Disk. We deploy TPC-W multi-tier web application benchmark on 4 processing nodes including 01 load balancer node (LB) based on NGINX[2] that receives requests from users and dispatches to web server with

[1]http://www.tpc.org/tpcw/
[2]https://www.nginx.com/

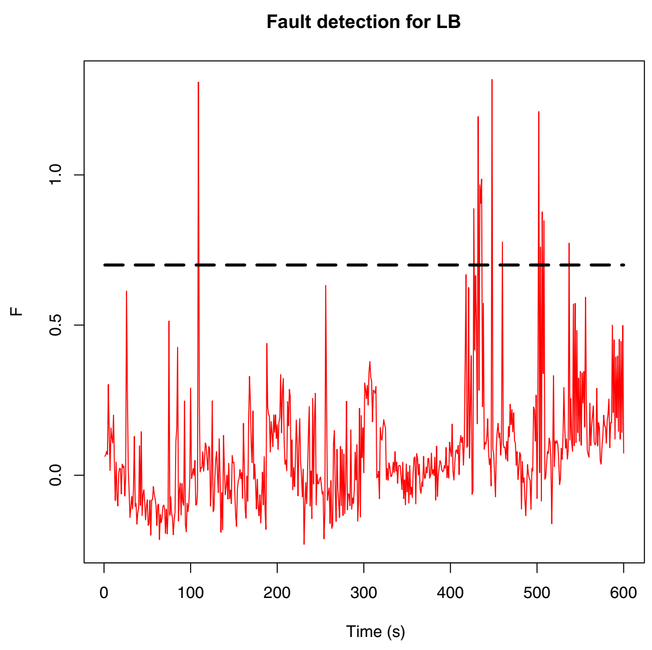

Fig. 5. Threshold-FOCSVM fault detection for LoadBalancer.

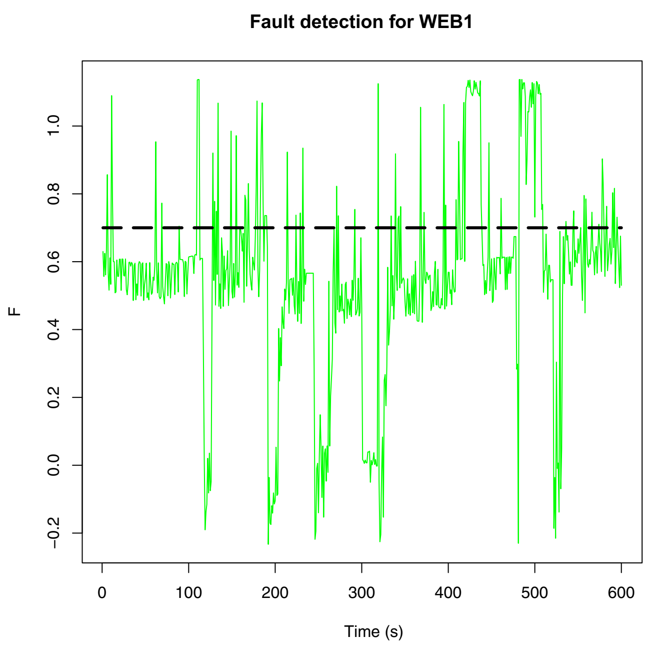

Fig. 6. Threshold-FOCSVM fault detection for WEB1.

RoundRobin load balancing algorithm; 02 nodes as web-server (Web1, Web2); 01 node serves as database server (DB). The system parameters of the processing nodes are monitored via the Prometheus[3] tool. Prometheus has a central component to process, called the Master Server, to collect data and query them, and a Node Exporter to export the system index follow the format of Prometheus.

The Prometheus monitoring tool is used to collect system parameter data for each processing node. Types of faults like [34] are put into the system with the method as in [35]–[37]. The system will be injected with two batches of faults including:

- Firstly, the faults are injected from 420th to 441st by increasing suddenly concurrent users from 20 to 200. This fault injection leads to the increase of Network traffic, lacking of Memory, and Disk I/O access.
- Secondly, from 485th to 512th, an endless loop in the source code of multi-tier application is triggered. This fault software leads to stack overflow.

B. Evaluating Fault Detection

In order to evaluate the proposed method, $Sensitivity$, $Specificity$, and $Accuracy$ are taken into consideration. Table 2 shows accuracy measurement of the model which are used to calculate these criteria. The formulas of these criteria are calculated as following:

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{44}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{45}$$

[3]https://prometheus.io/

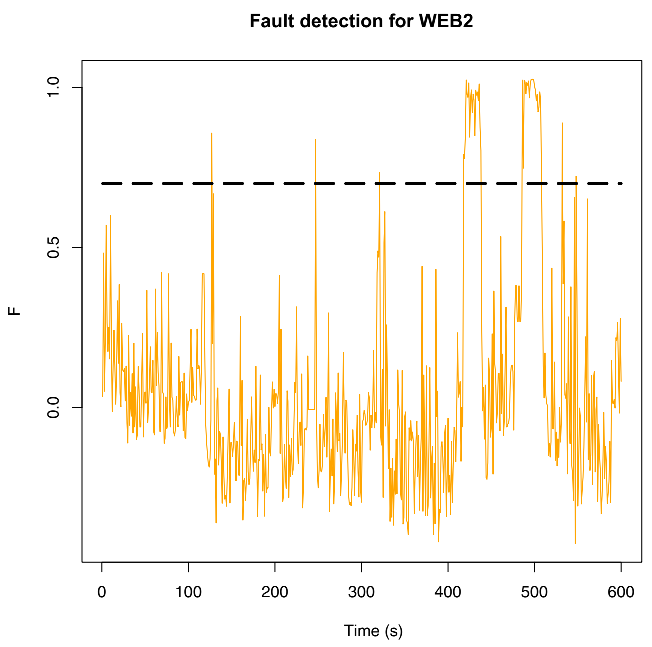

Fig. 7. Threshold-FOCSVM fault detection for WEB2.

$$\text{Accuracy} = \frac{TP}{TP + FP + TN + FN}, \tag{46}$$

where:

- TP: Shows the number of instances whose actual label is $Fault$ and the fault detection model also correctly recognizes this label.
- TN: Shows the number of instances whose actual label is $Normal$ and the fault detection model also correctly rec-

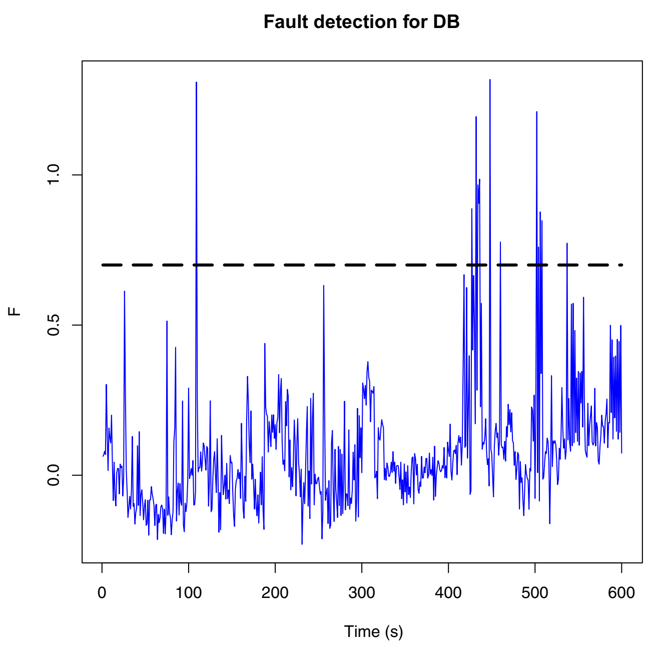

Fig. 8. Threshold-FOCSVM fault detection for DB.

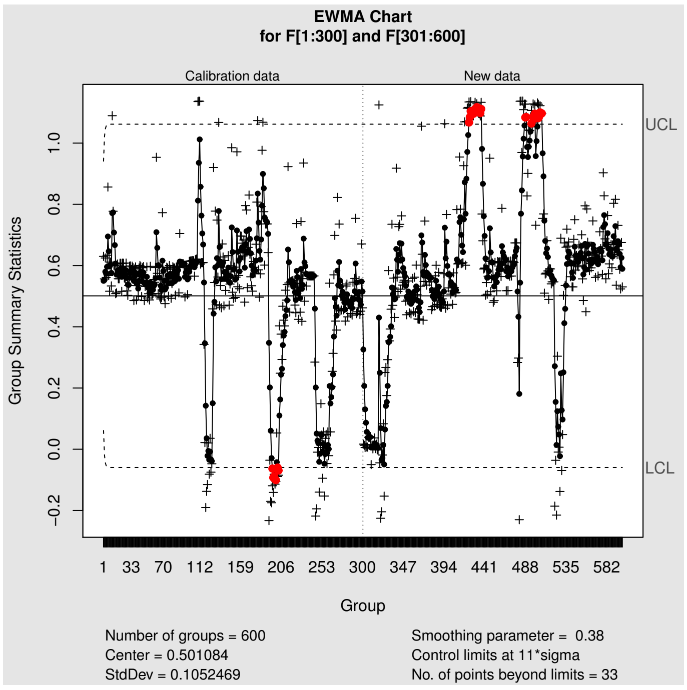

Fig. 10. EWMA-FOCSVM fault detection for WEB1.

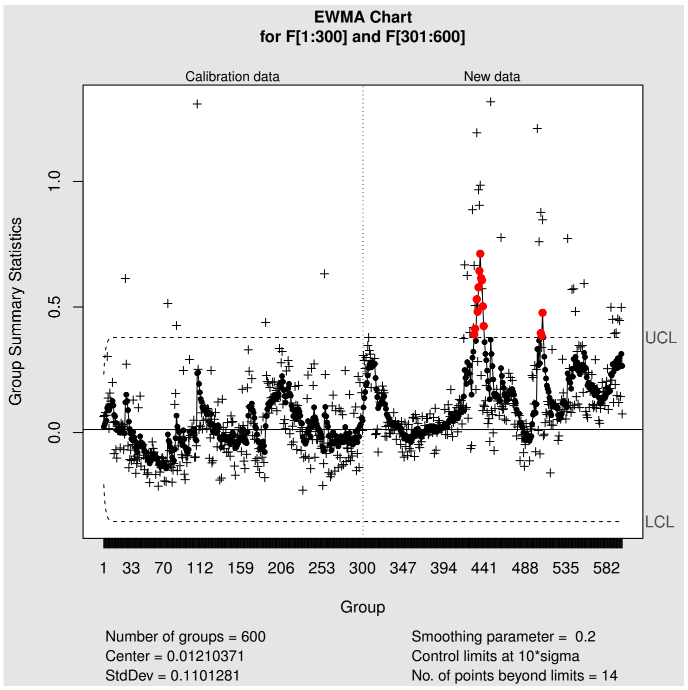

Fig. 9. EWMA-FOCSVM fault detection for Loadbalancer.

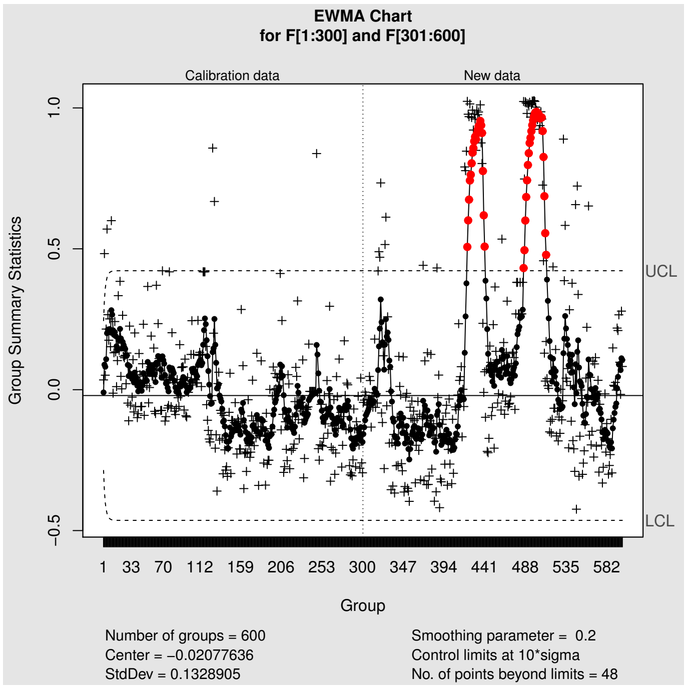

Fig. 11. EWMA-FOCSVM fault detection for WEB2.

ognizes this label.

- FP: Shows the number of instances whose actual data is $Normal$ but the fault detection model has mistakenly identified it as $Fault$.
- FN: Shows the number of records whose actual data is $Fault$ but the fault detection model has mistakenly identified it as $Normal$.

The value of decisive boundary $F(t)$ in (43) is used to detect faults by setting up a threshold in [15]. Figs. 5 to 8 show that the faults are detected when $F(x) > 0.7$. To evalutate the performance of Threshold-FOCSVM fault detection, the value of decisive boundary $F(t)$ are used as thresholds of receiver operating characteristic (ROC) curve as shown in Fig. 13. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. In machine learning, TPR is also known as sensitivity and FPR

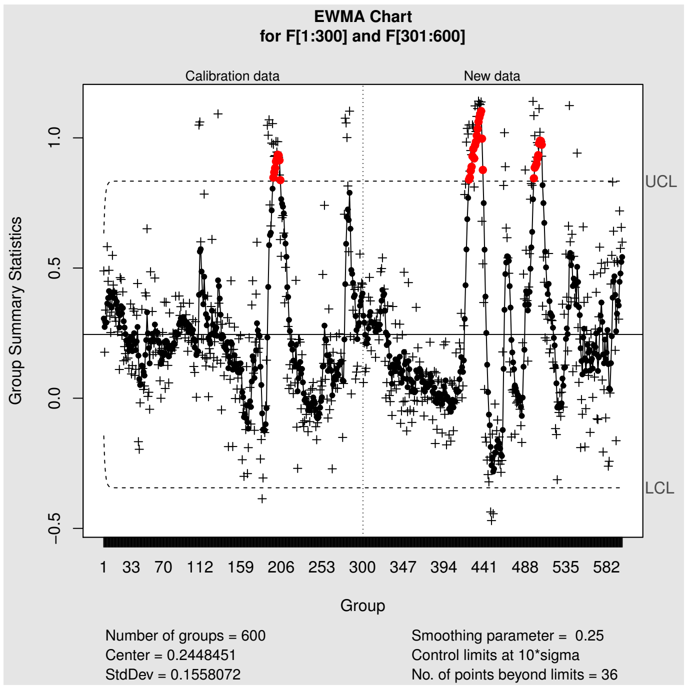

Fig. 12. EWMA-FOCSVM fault detection for DB.

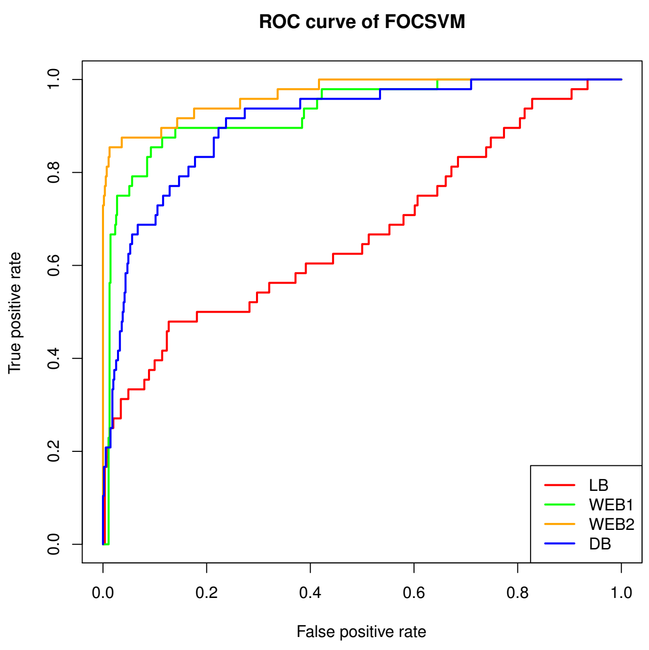

Fig. 13. ROC curve of fuzzy one-class support vector machine.

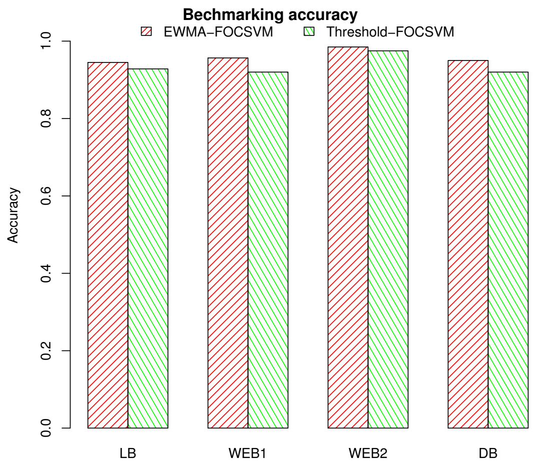

Fig. 14. Benchmarking accuracy of the fault detection model EWMA-FOCSVM and Threshold-FOCSVM.

is $(1 - Specificity)$. The loss caused by inaccurate fault detection of the prediction model plays key role in determining if $Sensitivity$ or $Specificity$ should be paid more attention to. In the fault detection model, these loss is counted by the cost of fault tolerance strategy in the later stage. A model with high $Sensitivity$ may prevent the system from faults and it is predicted to be false negative which cause lots of damage if the system possesses no solutions for fault tolerance. A model with high $Specificity$ allows to avoid cases of false positive in which the system has no fault while it is supposed to. Consequently, these false positives cause unnecessarily avoidable damages in the later stage of fault tolerance.

EWMA chart is used to monitor the fluctuation of Z_i in (34) and a fault alarm goes off when $Z_i < LCL_{x(i)}$ or $Z_i > UCL_{x(i)}$. As shown from Figs. 9 to 12, the results demonstrate that the values of decisive boundary function in (41) are nearly between the LCL and UCL of EWMA control chart when the faults are not triggered. However, to those which are over the UCL of EWMA are classified into faults. For example, the faults are triggered in the 420th sample, but the fault detection is raised in the 423rd for the first fault injection which lasts from 420th to 441st. Similarly, fault detection is raised in the 488th sample for the second fault injection (from 485th to 512th).

Fig 14 illustrates that the fault detection accuracy of EWMA-FOCSVM and the thresholds of Threshold-FOCSVM. EWMA-FOCSVM sees higher results in all criteria compared including LB, WEB1, WEB2 and DB.

C. *Evaluating Fault Diagnosis*

At the first process, the suddenly increasing of concurrent users causes a shortage of memory in both LB and DB, network faults in WEB 1 and Context Switch Faults at WEB 2. At the second fault injection, the endless loop fault of multi-tier application causes CPU hog in both LB and WEB 2, I/O fault in WEB 1 and lack of memory in DB. Both of two case, memory is an importance resource in DB. Therefore, it can be concluded that, this type of fault due to lack of resources requires proper VM types for multi-tier application to meet the quality of service (QoS).

The fault diagnosis algorithm is evaluated by calculating the errors of RF predictor which is calculated as following:

$$Error = \frac{1}{N_{OOB}} \sum_{i=1}^{N_{OOB}} \mathcal{E}(Y, \hat{Y}_{OOB}), \tag{47}$$

where N_{OOB} is OOB's sample size, $\mathcal{E}$ is an error function, $\hat{Y}_{OOB}$ is calculated in (37). Fig. 15 illustrates the decreasing of

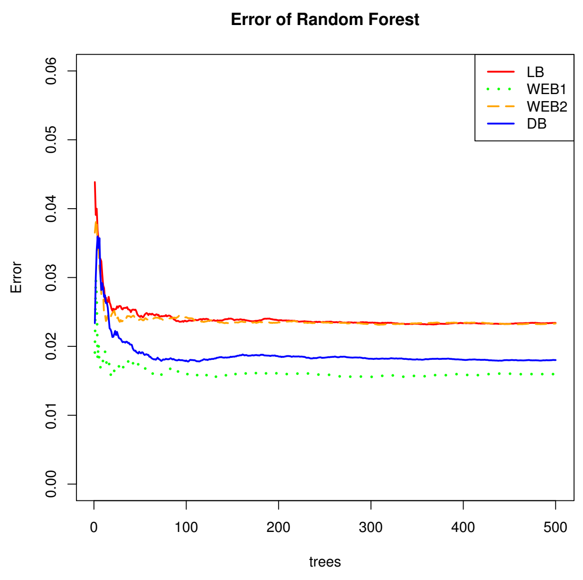

Fig. 15. Random Forest algorithm errors correspond to the trees quantity.

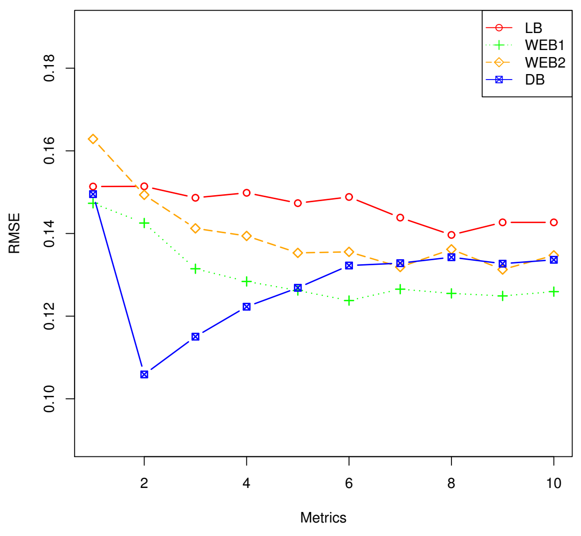

Fig. 17. Root Mean Square Error of RFE-BDT.

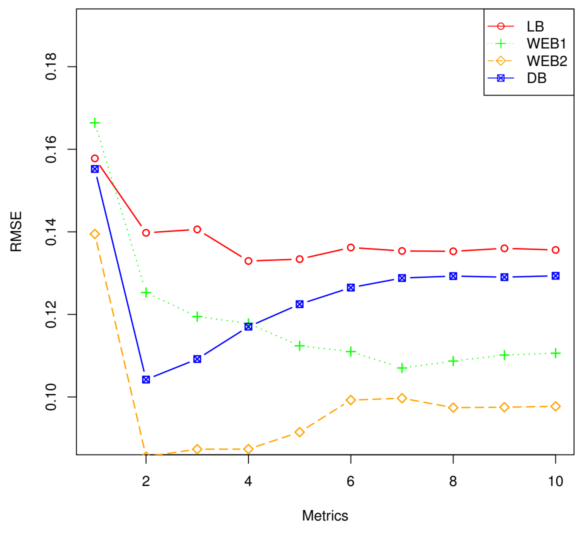

Fig. 16. Root Mean Square Error of RFE-RF.

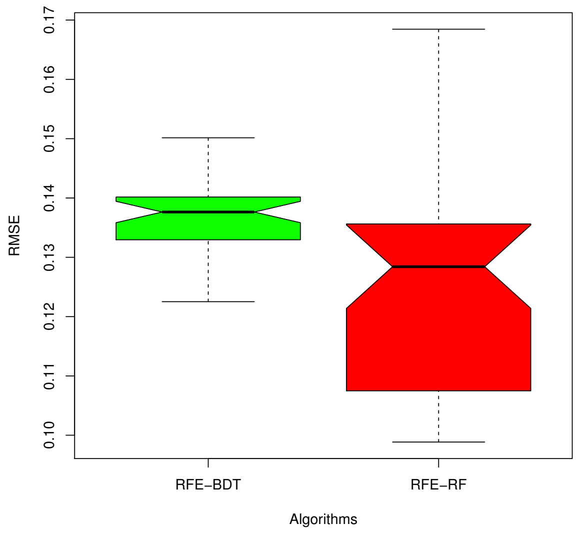

Fig. 18. Root Mean Square Error's Boxplot of RFE-RF and RFE-BDT.

errors in Random Forest algorithm when increasing trees. The proportion of errors has high fluctuation in the fifty first trees. The error rate keeps stable when trees reach from 100. And, the error is almost convergent in 500 trees.

The model fit is evaluated by using Root Mean Square Error (RMSE) which is calculated as following:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Y_{\text{Predicted}} - Y_{\text{Actual}})^2}{N}}, \tag{48}$$

where N is sample size.

The Bagged decision trees algorithm [44], [45] which is among ensemble-based algorithms like as RF algorithm is applied to a baseline algorithm. The Bagged decision trees are bagged ensembles where each model is a decision tree. Compared to Bagged decision trees which require all features for a node splitting, Random forests split features into subsets randomly and the best split subset is selected for each node splitting. The unbalanced dataset which instances are labeled by using EWMA-FOCSVM fault detection method in the two injection faults is used to compare performance of Random Forest and Bagged Decision Trees. Figs. from 16 to 17 show the results of RFE-RF and RFE-BDT method which present the relationship between RMSE. The RMSE minimum values of REF-RF are lower than REF-BDT at each metric of LB, WEB1, WEB2, and DB. As shown in Fig. 18, generally, the median RMSE of

RFE-RF is lower than of RFE-BDT. However, the range RMSE of RFE-RF is larger than RFE-BDT. Therefore, the exploration ability of Random Forest is better than Bagged Decision Trees and proves the robust of Random Forest against unbalanced training data.

V. CONCLUSIONS

In this paper, a fault detection and diagnosis approach is proposed for multi-tier web application in infrastructure cloud computing. This problem is addressed through three-step approach, where we first proposed the Fuzzy one-class support vector machine model to detect abnormal data based on the decisive boundary. Then, the exponentially weighted moving average chart which monitors the decisive boundary value of FOCSVM model is applied to detect faults for multi-tier web application. Finally, the Random Forest ranking feature algorithm is applied to determine if suspicious metrics are the root of faults. The performance fault detection and diagnosis approach is evaluated by comparing it with baseline methods. In fault detection problem, the accuracy of EWMA-FOCSVM is higher than one in Threshold-FOCSVM. In order to compare Random Forest algorithm and Bagged Decision Trees in term of ranking features, Recursive Feature Elimination method with each iteration using Random Forest or Bagged Decision Trees is applied. The experiment result shows the Root Mean Square Error values of Recursive Feature Elimination with Random Forest algorithm is lower one in Bagged Decision Trees. Also, the exploration ability of Random Forest algorithm in unbalanced training data is higher Bagged Decision Trees. However, more study should be done on byzantine fault for further research.

REFERENCES

[1] A. D. Josep, R. Katz, and A. Konwinski, G. Lee, D. Patterson, A. Rabkin, A. Rapkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *ACM Commun.*, vol. 53, no. 4, pp. 50–58, 2010.

[2] D. Oppenheimer, A. Ganapathi, and D. .A. Patterson, "Why do Internet services fail, and what can be done about it?," in *Proc. USENIX USITS*, 2003.

[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surveys*, vol. 41, no. 3, pp. 15, 2009.

[4] E. Sindrilaru, A. Costan, and V. Cristea, "Fault tolerance and recovery in grid workflow management systems," in *Proc. CISIS*, 2010, pp. 475–480.

[5] Y. Zhang, A. Mandal, C. Koelbel, and K. Cooper, "Combined fault tolerance and scheduling techniques for workflow applications on computational grids," in *Proc. IEEE/ACM CCGRID*, 2009, pp. 244–251.

[6] D. C. Plummer, T. J. Bittman, T. Austin, D. W. Cearley, and D. M. Smith, "Cloud computing: Defining and describing an emerging phenomenon," *Gartner*, vol. 17, June 2008.

[7] G. Jiang, H. Chen, and K. Yoshihira, "Modeling and tracking of transaction flow dynamics for fault detection in complex systems," *IEEE Trans. Dependable Secure Comput.*, vol. 3, no. 4, pp. 312–326, 2006.

[8] H. Kriegel, P. Kröger, J. Sander, A. Zimek,, "Density-based clustering," *Wiley Interdisciplinary Reviews: Data Mining Knowledge Discovery*, vol. 1, no. 3, pp. 231–240, 2011.

[9] B. Tang and H. He, "A local density-based approach for outlier detection," *Neurocomputing*, vol. 241, pp. 171–180, 2017.

[10] S. Agrawal, J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Comput. Science*, vol. 60, pp. 708–713, 2015.

[11] J. Han, J. Pei, and M. Kamber, "Data mining: Concepts and techniques," 2011.

[12] J. C. Platt **et al.**, "Estimating the support of a high-dimensional distribution", *Technical Report MSR-T R-99–87, Microsoft Research (MSR)* 1999.

[13] D. M. Tax and R. P. Duin, "Support vector data description", *Machine learning* vol. 54, no. 1, pp. 45–66, 2004.

[14] Q. Leng, H. Qi, J. Miao, W. Zhu, and G. Su, "One-class classification with extreme learning machine," *Mathematical problems engineering*, 2015.

[15] S. Yin, X. Zhu, and C. Jing, "Fault detection based on a robust one class support vector machine," *Neurocomputing*, vol. 145, pp. 263–268, 2014.

[16] R. Jhawar and V. Piuri, "Fault tolerance and resilience in cloud computing environments," *Comput. Information Security Handbook*, pp. 165–181, 2017.

[17] A. Tchernykh, U. Schwiegelsohn, V. Alexandrov, and E. Talbi, "Towards understanding uncertainty in cloud computing resource provisioning," *Procedia Comput. Science*, vol. 51, pp. 1772–1781, 2015.

[18] T. Wang, W. Zhang, C. Ye, J. Wei, H. Zhong, and T. Huang, "FD4C: Automatic fault diagnosis framework for Web applications in cloud computing," *IEEE Trans. Systems Man Cybernetics: Systems*, vol. 46, no. 1, pp. 61–75, 2015.

[19] T. Wang, J. Xu, W. Zhang, Z. Gu, H. Zhong, "Self-adaptive cloud monitoring with online anomaly detection," *Future Generation Computer Systems*, vol. 80, pp. 89–101, 2018.

[20] A. Tchana, L. Broto and D. Hagimont, "Fault tolerant approaches in cloud computing infrastructures," in *Proc. ICAS)*, pp. 42–48, 2012.

[21] M. Lin, Z. Yao, F. Gao, and Y. Li, "Data-driven Anomaly Detection Method for Monitoring Runtime Performance of Cloud Computing Platforms," *Int. J. Hybrid Inf. Technol.*, vol. 9, no. 2, pp. 439–450, 2016.

[22] J. Pinto, P. Jain, and T. Kumar, "Hadoop distributed computing clusters for fault prediction," in *Proc. ICSEC*, 2016, pp. 1–6.

[23] M. Smara, M. Aliouat, A. K.. Pathan, and Z. Aliouat, "Acceptance test for fault detection in component-based cloud computing and systems," *Future Generation Comput. Systems*, vol. 70, pp. 74–93, 2017.

[24] C. Pham, L. Wang, B. C. Tak, S. Baset, C. Tang, Z. Kalbarczyk, and R. K. Iyer, "Failure diagnosis for distributed systems using targeted fault injection," *IEEE Trans. Parallel Distributed Systems*, vol. 28, no. 2, pp. 503–516, 2016.

[25] P. Zhang, S. Shu, and M. Zhou, "An online fault detection model and strategies based on SVM-grid in clouds," *IEEE/CAA J.Automatica Sinica*, vol. 5, no. 2, pp. 445–456, 2018.

[26] T. Wang, J. Wei, W. Zhang, H. Zhong, and T. Huang, "Workload-aware anomaly detection for web applications," *J. Systems Software*, vol. 89, pp. 19–32, 2014

[27] T. Wang, W. Zhang, C. Ye, J. Wei, H. Zhong, and T. Huang, "FD4C: Automatic fault diagnosis framework for Web applications in cloud computing," *IEEE Trans. Systems, Man, Cybernetics: Systems*, vol. 46, no. 1, pp. 61–75, 2015.

[28] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, 2001.

[29] C.-F. Lin and S.-D. Wang, "Fuzzy support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 464–471, 2002.

[30] M. Liu, B. C. Vemuri, S.-I. Amari, and F. Nielsen, "Total Bregman divergence and its applications to shape retrieval," in *Proc. IEEE CVPR*, 2010, pp. 3463–3468.

[31] S. Yin and G. Wang, "A modified partial robust M-regression to improve prediction performance for data with outliers," in *Proc. IEEE ISIE*, 2013, pp. 1–6.

[32] J. Jiong, Z. Hao-ran, "A fast learning algorithm for One-Class Support vector machine," in *Proc.ICNC*, 2017, pp. 19–23.

[33] T. M. Khoshgoftaar, M. Golawala, and J. V. Hulse, "An empirical study of learning from imbalanced data using random forest," in *Proc. IEEE ICTAI*, 2007, pp. 310–317.

[34] , S. Pertet and P. Narasimhan, "Causes of failure in web applications," *Technical Report CMU-PDL-05-109, Carnegie Mellon University*, 2005.

[35] G. Casale, N. Mi, and E. Smirni, "Model-driven system capacity planning under workload burstiness," *IEEE Trans. Comput.*, vol. 59, no. 1, pp. 66–80, 2009.

[36] H. Kang, H. Chen, and G. Jiang, "PeerWatch: a fault detection and diagnosis tool for virtualized consolidation systems," in *Proc. ICAC*, 2010, pp. 119–128.

[37] G. Jiang, H. Chen, K. Yoshihira, and A. Saxena, "Ranking the importance of alerts for problem determination in large computer systems," *Cluster Comput.*, vol. 14, no. 3, pp. 213–227, 2011.

[38] Sematech, NIST *NIST SEMATECH*, 2006.

[39] A. Sîrbu and O. Babaoglu, "Towards data-driven autonomics in data centers," in *Proc. ICCAC*, 2015, pp. 45–56.

[40] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications", *Expert Systems Applications*, vol. 73, pp. 220–239, 2017.

[41] A. Javadpour, S. K. Abharian, and G. Wang, "Feature selection and intrusion detection in cloud environment based on machine learning algorithms", in *Proc. IEEE ISPA/IUCC*, 2017, pp. 1417–1421.

[42] B. F. Darst, K. C. Malecki, and C. D. Engelman, “Using recursive feature elimination in random forest to account for correlated variables in high dimensional data”, *BMC genetics*, vol. 19, no. 1, p. 65, 2018.
[43] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, “Gene selection for cancer classification using support vector machines”, *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002.
[44] I. Guyon, S. Gunn, M. Nikravesh, L. A. Zadeh, “Feature extraction: foundations and applications”, 2008.
[45] M. A. Munson and R. Caruana, “Joint European Conference on Machine Learning and Knowledge Discovery in Databases”, *Machine learning*, vol. 46, pp. 144–159, 2009.

Khiet Thanh Bui received B.Sc. degree on Software Engineering from Ho Chi Minh of Technology in 2010. He acquired his Master’s degree from Posts and Telecoms Institute of Technology in Ho Chi Minh in 2012. He is working at Faculty of Engineering-Technology, Thu Dau Mot University as a lecture. At present, he is a Ph.D candidate at Computer Science, Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, VNU-HCM. His research focuses on cloud computing.

Len Van Vo received B.Sc. degree on computer science from VNUHCM-University of Science in 2015. He is studying for a master’s degree at Faculty of Engineering -Technology, Thu Dau Mot University, Binh Duong, Vietnam. His research area is fault detection in cloud computing

Canh Minh Nguyen received B.Sc. degree on Information Technology from SaiGon University in 2011. At present, he works at Key Learning Facilities Center in Saigon university and is studying for a master’s degree. His research focus includes cloud computing security and approximate algorithms.

Tran Vu Pham is an Associate Professor and also the Dean of the Faculty of Computer Science and Engineering, Ho Chi Minh City of Technology, VNU-HCM, Vietnam. He is interested in developing and applying new and advanced techniques and tools from big data, IoT, and distributed systems to solve real life problems in urban traffic, smart cities, agriculture, etc. Tran Vu Pham received his PhD degree in Computing from the University of Leeds, UK.

Hung Cong Tran received the master of Engineering degree in Telecommunications Engineering course from postgraduate department Hanoi University of technology in Vietnam, 1998. He received Ph.D at Hanoi University of technology in Vietnam, 2004. His main research areas are B – ISDN performance parameters and measuring methods, QoS in high speed networks, MPLS, Wireless Sensor Network, Cloud Computing. He is, currently, an Associate Professor PhD. of Faculty of Information Technology II, Posts and Telecoms Institute of Technology in Ho Chi Minh, Vietnam.