Automated Modeling of Real-Time Anomaly Detection using Non-Parametric Statistical Technique for Data Streams in Cloud Environments

Smrithy G S and Ramadoss Balakrishnan, *Member, IEEE*

Original scientific paper

***Abstract*—Online anomaly detection plays a vital role in improving the performance of cloud data centers by identifying unusual behaviors. In this paper, we propose an online anomaly detection framework using non-parametric statistical technique in cloud data center. The major advantage of the proposed work is its capability of automatic re-computing of the model, according to the fundamental changes in the data. In order to determine the accuracy of the proposed work, we experiment it to data collected from RUBis cloud testbed (Dataset 1) and Yahoo Cloud Serving Benchmark (YCSB) (Dataset 2). Our experimental results show the greater accuracy in terms of True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR) and False Negative Rate (FNR).**

***Index Terms*—Algorithm, Automated Modeling, Cloud Data Center, Data Stream, Non-Parametric Statistical Technique, Online Anomaly Detection.**

I. INTRODUCTION

Cloud computing service providers built data centers that contain hundreds of thousands of servers [1]. The size and complexity of cloud data centers are expected to grow further as more and more services are migrating to cloud platforms. They are increasingly characterized by extremely large scale and complexity. Utility based cloud services [2] like Amazon web services, SQL server data services and Google App are able to serve thousands of thousands businesses to run their own applications, each of which having varying workload characteristics. Thus it is very difficult to supervise cloud data centers, and the failure to do so will in turn lead to massive negative impact in profit on account of inadequacy of responsiveness and availability. Therefore there is a need to improve cloud data center performance by establishing frameworks for detecting problems (that deviate from normal characteristics) in real time. It includes collecting online data streams, analyzing it and taking remedial actions when needed. In essence, the system should aim to capture unusual or anomalous system behavior in real time. It is a critical task for detecting/identifying atypical characteristics when large amount of data streams are generated continuously in cloud data centers. The key element of this paper is online anomaly detection in order to improve the cloud data center performance.

Online anomaly detection refers to identifying abnormal/unusual behavior that deviates significantly from expected usual behavior in real time. Anomaly detection techniques can be broadly classified into different categories such as data mining based, machine learning based, cognition based, statistical based etc. Here we opt for statistical based online anomaly detection. Among that, statistical techniques can be categorized as parametric, non-parametric and semi-parametric techniques [3]. Parametric techniques assume the knowledge of underlying distribution based on a fixed set of parameters. Some parametric statistical techniques are Gaussian model-based, regression model-based, mixture of parametric distributions-based etc. [4]. Non-parametric techniques make no assumptions on the probability distribution of the data. Some examples of non-parametric techniques are histogram based and kernel based techniques. Semi-parametric techniques are a combination of parametric and non-parametric techniques to build, fit and validate statistical models. In this paper, we utilize non-parametric statistical technique for anomaly detection due to the following reasons. i) It makes less number of assumptions ii) It is easier to compute iii) It can be used with all scales.

It is evident that in a real world cloud data center [5], 84% of the virtual machines show spikes in their CPU utilization at least once in a month [6]. From the above fact, we can see that it is hard to classify a false alarm based on the dynamic nature of the workload characteristics. Few drawbacks of such false alarms are unwanted inspections, unnecessary disruption of users applications etc.

Manuscript received February 10, 2019; revised April 9, 2019. Date of publication July 5, 2019. Date of current version July 5, 2019. The associate editor Toni Mastelić has been coordinating the review of this manuscript and approved it for publication.

Authors are with the Department of Computer Applications, National Institute of Technology, Tiruchirappalli, TamilNadu, India (corresponding author, e-mails: smrithygs1990@gmail.com, brama@nitt.edu).

Digital Object Identifier (DOI): 10.24138/jcomss.v15i3.717

A good anomaly detection system should be capable of detecting anomalies by considering the dynamic nature of workloads characteristics in cloud data centers since the pattern of workload may show hourly, daily, weekly or monthly changes. A system that uses static threshold for anomaly detection is only valid for a short span of time. Another fact is, classifying individual data points as anomalies can bring about false alarm rates. Thus there is a need in art for a system which can show significant results and lower false alarm rates in the scenario of dynamic nature of the loads. Hence the motivation of this paper is to achieve high accuracy in identifying anomalies in cloud data centers having dynamic workloads by lowering the false alarm rates. Therefore, the main focus of this paper is using non-parametric statistical technique for anomaly detection in real time data streams in cloud data centers.

The contributions of this paper are highlighted as follows:

1. We have developed an online anomaly detection system that is capable of detecting anomalies in real time data stream in cloud data centers. This system uses a non-parametric statistical test, in particular, chi-square two sample test for comparing two data samples. Here we are capturing data stream as a time window at specific intervals. The system is designed for detecting abnormalities on daily basis in regular time intervals. It can also be extended to work on weekly or monthly basis of data patterns. The system initially builds the base model using data without any major security events. The framework is capable of automatically rebuilding the model when there is a significant change in the fundamental pattern of data. The proposed system has low complexity and is scalable to process large amounts of data.
2. We have experimented the accuracy of the proposed algorithm using two different datasets; Dataset 1 and Dataset 2. Dataset 1 and Dataset 2 are collected form RUBiS cloud testbed [7] and Yahoo Cloud Serving Benchmark (YCSB) [8] respectively. Our experimental results show a greater accuracy in terms of True Positive Rate, False Positive Rate, True Negative Rate and False Negative Rate.

The rest of this paper is organized as follows. Section II covers the related work. In section III preliminaries are shortly presented. In section IV the proposed online anomaly detection algorithm using non-parametric statistical technique is described in detail. Section V presents the experimental results. Section VI concludes the paper.

II. Related Work

The goal of anomaly detection problems is to identify data patterns that do not conform to a certain expected pattern in a dataset. To design a framework that detects anomalies, a statistical model is fit for normal behavior and then an appropriate inference test is applied to check whether the new observed instance belongs to this model. In order to solve the anomaly detection problems [9], many parametric techniques such as Tukey limits, ANOVA tests, Pearson correlation, Grubb's test, Student-t test etc are used. However, since the ultimate goal is to detect anomalies, learning the distributions first and then constructing the detection rules may not yield optimal performance. It is thus desirable to design data-driven nonparametric tests, which directly perform anomaly detection using data without estimating the distributions as an intermediate step. Furthermore, since such tests do not exploit any information about the distributions, they can be designed to provide universal performance guarantee for arbitrary distributions.

An anomaly detection technique that uses non-parametric statistical techniques makes fewer or no assumptions of the given data, such as smoothness of density. A simple non-parametric based anomaly detection system can use histogram-based technique. Kruegel et al. [10], [11] used histogram-based technique for detecting web based attacks and fraud detection respectively. A more versatile technique under non-parametric approach for detecting anomalies is based on parzen window estimation [12], [13]. It includes approximating the actual density using kernel functions.

TABLE I
Statistical Approaches for Anomaly Detection

Types	Approach
Parametric	Mixture Models 1. Gaussian Mixture Model (GMM) [14] Extreme Value Theory (EVT) [15] Autoregressive Integrated Moving Average (ARIMA) [16] State-Space Models 1. Hidden Markov Model (HMM) [17] 2. Kalman filter [18]
Non-Parametric	Kernel Density/ Parzen Window Estimators [19] Negative Selection [20]

Numerous commercial studies that use more refined statistical methods for anomaly detection have been proposed [21] -[25]. Despite a few of the methods can be applied for real time anomaly detection in cloud environment because of their high computational overhead. Table I shows different statistical approaches for anomaly detection. Buzen et al. [26] proposed an anomaly detection technique that enhances the fixed threshold strategies which repeatedly calculates the control limits in an iterative manner when a new data arrives. Datar et al. [27] presented a method for managing and aggregating data streams in a sliding window. They suggested their proposed model to be applied in areas like telecommunications where data are generated continuously. In such applications, instead of processing the whole historical data, we can focus on the most recent set of observations.

As evident from the above literature, none of the researchers have addressed the construction of automated modeling (considering dynamic changes in the data) using non-parametric statistical techniques for detecting anomalies in real time cloud environment. The proposed anomaly detection

framework uses non-parametric statistical technique for detecting anomalous characteristics based on performance metrics such as CPU utilization. The strength of the system lies in its capability of automatic re-computing of the model, according to the fundamental changes in the data. This improves the precision of anomaly detection and in turn reduces false alarm rate.

III. Chi-square Test

The main objective of statistical anomaly detection technique is to determine if there is any significance difference between the distribution of the collected set of sample data and the distribution of the normal data. In short, we have to check whether the two data sets are drawn from the same population distribution function. The task will be easy when the data distribution is known a priori. Nonetheless we cannot assume that the data should follow a particular distribution in practical scenarios. Thus there is a need to check the samples against the samples that were previously accepted as normal data. The techniques used for such situations are known as non-parametric statistical techniques. For our problem, we are using a particular non-parametric statistical technique called chi-square test. It is used to check how much the observed values of a particular given sample significantly differ from the expected values of the distribution [28]. It shows how well an experimental data fits an expected probability distribution. Unlike tests like Z and t, chi-square test theories about the full distribution instead of any single statistic from the distribution [29].

Chi-square test can be used to compare between two datasets that are quantized into a specific number of bins [30]. Binning can be done by grouping the events into specific ranges of the variable such as 0-10, 10-20, 20-30 etc. For our problem, as we are taking the percentage of CPU utilization which can take values between 0 and 100 as the input of our anomaly detection framework, we quantize it into 20 bins of equal size. Given two binned datasets that are of equal size, let P_i be the number of observations for the first dataset and let Q_i be the number of observations in the second dataset in bin i, then the chi-square statistics for comparing both the binned datasets is given by,

$$\chi^2 = \sum_i \frac{(P_i - Q_i)^2}{P_i + Q_i} \qquad (1)$$

The number of degrees of freedom is equal to one less than the number of bins, if the sum of P_i's is undoubtedly equal to sum of Q_i's. Otherwise the number of degrees of freedom will be equal to the number of bins [31]. In real time, when data streams that are arriving in a particular time interval are collected using a time window, there are possibilities that the number of data instances in one time window differs from the number of data instances in another time window. The possible reasons may be practical constraints such as network delays etc. In such scenarios the anomaly detection framework should be able to compare two different sized windows, i.e. it should be able to compare datasets of unequal sizes. For unequal number of data points, the corresponding equivalent of Eqn (1) can be written as [30],

$$\chi^2 = \sum_i \frac{\left(\sqrt{\frac{Q}{P}}P_i - \sqrt{\frac{P}{Q}}Q_i\right)^2}{P_i + Q_i} \qquad (2)$$

where $P \equiv \sum_i P_i$ and $Q \equiv \sum_i Q_i$ are the number of data instances respectively.

The computed test statistic χ^2 will be compared against a threshold. For datasets of unequal sizes, the threshold will be commonly set to the point in the chi-squared cumulative distribution function with degrees of freedom equal to number of bins that conforms to a 0.95 or 0.99 confidence level [31]. Any statistical tools like R [32], SAS [33] etc. can be used to fix this threshold. If the computed test statistic is lesser than the threshold, we can say that the observed dataset follows the same distribution of the normal dataset; otherwise we can say that the distributions are different.

IV. Proposed Work

The proposed algorithm focuses on finding anomalous pattern in a collection of data instances rather than finding individual anomalous data instances to reduce false alarm rates. The system collects data in particular time intervals using a time window. If the distribution of the data instances in the window varies significantly from the base model, the window will be declared anomalous. Fig. 1 shows the flowchart of the proposed anomaly detection framework for detecting anomalous data streams in cloud data centers. For data streams, systems that update and decay over time is recommended. Hence, our proposed model is capable of automatic updating when there is a fundamental change in the pattern of data. The proposed work consists of three folds. Initially, the procedure $BuildModel$ builds the base model using the performance metric, then the algorithm $FindDistribution$ finds the unknown distribution of the data and finally the algorithm $Online - Anomaly - Detection$ detects abnormal characteristics in the performance data.

The proposed framework is described as follows. We are dividing a day data into n equal time interval windows. Initially the Algorithm 1 builds the base model using the data of interest, here CPU utilization. Here the data streams are collected as time windows. For example, data stream collected at i^{th} time interval will be stored in W_i. The data used for building the base model should be from a clean day which does not have any major security events. But to avoid the degradation of model accuracy, the algorithm first eliminates any possible outliers before building the model. This is shown in steps from 5 to 8. After eliminating the outliers the clean window is represented as Wp_i. The base model for the i^{th} time interval will be generated using Wp_i. The distribution bm_i for Wp_i is computed using the algorithm 2. BM stores the distributions bm_i; such that $i \leq n$ where n is the number of

time intervals. Finally the Algorithm 1 returns BM.

The Algorithm 2 describes how to generate the distribution of a data stream which is collected as windows. The inputs of the algorithm are window W, and number of bins NOB. The procedure find the unknown distribution of window W using a bin-based histogram approach from steps 3 to 7. In window W, each data instance is called a tuple. From the tuple, a data point dp with its value is extracted. Then Bin Index, $B_{current}$ as shown in step 4. Here W_{min} and W_{max} are the lower bound and upper bound of window W respectively. It can be calculated experimentally. As we have used CPU utilization percentage for our proposed work, its lower and upper bound are 0 and 100 respectively. The algorithm then emits $B_{current}$ with its frequency as a tuple and calculates each bin frequency M_i. It finally returns Empirical Frequency $\widehat{M}$.

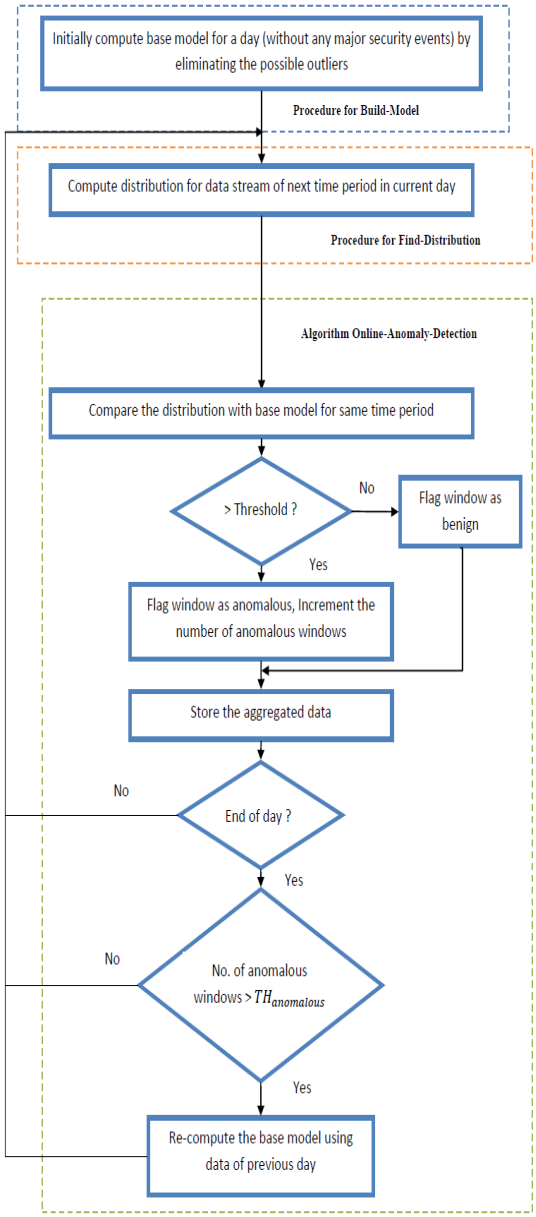

Fig. 1. Flow chart of the proposed anomaly detection framework.

Algorithm1: Procedure for Build-Model

1 Procedure $BuildModel(DayData)$
2 Begin
3 $W_i \leftarrow$ data collected at i^{th} time interval
4 For $i \leftarrow 1$ to n do
5 Return the set of values of W_i that falls beyond $\pm 3\sigma$
6 For the set of values returned in step 5 extract the set of values that exceeds $\pm 3\sigma$
7 Label the set of values returned in step 6 as outliers
8 Return the remaining value of W_i as Wp_i
9 $bm_i \leftarrow FindDistribution(Wp_i, NOB)$
10 Add to BM; bm_i
11 End
12 Return BM
13 End

Algorithm 2: Procedure for Find-Distribution

1 Procedure $FindDistribution(Window\ W, NOB)$
2 Begin
3 Take each instance of W called tuple and extract a data point, dp with its value from this tuple
4 Generate Bin Index, $B_{current}$

$$B_{current} = \frac{(dp - \mathrm{W_{min}})}{(\mathrm{W_{max}} - \mathrm{W_{min}})} * \mathrm{NOB}$$

5 Return $B_{current}$ with its frequency as a tuple
6 Count the total number of frequencies for each bin, M_i, $\forall i = 1,2 \ldots, NOB$
7 Return Empirical Frequency $\widehat{M}$
8 End

Algorithm 3: Online-Anomaly-Detection

$Input: DS, BM, TH, N_{anomalous} = 0, TH_{anomalous}$

1 Begin

2 $W_i \leftarrow$ data collected at i^{th} time interval

3 For $i \leftarrow 1\ to\ n$ do

4 $current_i \leftarrow FindDistribution(W_i, NOB)$

5 Compute the test statistic,

$$\chi^2 \leftarrow \sum_{j=1}^{NOB} \frac{(\sqrt{\frac{bm_i}{current_i}} * c_j - \sqrt{\frac{current_i}{bm_i}} * b_j)^2}{c_j + b_j}, \text{ where}$$

$bm_i = \sum_{j=1}^{NOB} b_j$ and $current_i = \sum_{j=1}^{NOB} c_j$

6 If $\chi^2 < TH$

7 Flag window W_i as benign

8 Else

9 Flag window W_i as anomalous

10 $N_{anomalous} = N_{anomalous} + 1$

11 $DataCurrent \leftarrow$ Store aggregated volume of data

12 End

13 If $N_{anomalous} < TH_{anomalous}$

14 Retain BM for next day anomaly detection

15 Else

16 Re-compute $BuildModel(DataCurrent)$

17 End

The Algorithm 3 illustrates our proposed anomaly detection framework. Here we have focused more on a set of data points in a particular time interval called window rather than an individual data point. Data collected at i^{th} time interval will be stored in W_i. In step 4, the distribution of window W_i is computed and represented as $current_i$. Then it compares $current_i$ with the baseline model of same time period bm_i using chi-square test statistic as shown in step 5. Here b_j represents the number of items in bin j for the base model dataset at time window i and c_j represents the number of items in bin j for the current data stream at time window, i.

If the test statistic does not exceeds a threshold, the window W_i is declared as benign, otherwise as anomalous and an alarm is raised. Subsequently the anomalous counter, $N_{anomalous}$ is also incremented and the aggregated data stream is stored as DataCurrent. At this stage, a determination is made whether the day has ended. If the day has not ended, the algorithm goes for a loop from step 3 to 11. Otherwise, the algorithm checks whether the anomalous counter value has exceeded a particular threshold $TH_{anomalous}$. For example, if the algorithm declares half of the number of windows to be abnormal, then there might be some fundamental change in the data to which the framework needs to adapt. If it has not exceeded, then the same set of baseline models are retained. Otherwise, the algorithm rebuild the baseline models using the data of the previous day; DataCurrent.

V. Experiments and Results

The accuracy of the proposed algorithm is evaluated using two datasets, Dataset 1 and Dataset 2. The details of experimental set up and results are described in the following section.

A. Experimental Setup

For evaluating the proposed algorithm, for the first scenario we have established a RUBiS cloud testbed [7] with two compute nodes: H1 and H2. RUBiS is an open source cloud based auction site benchmark. Its services are deployed in virtual machines mapped to the cloud's machine resources when used in cloud environment. The testbed uses 5 virtual machines (VM 1 to VM 5) on Xen Platform hosted on H1 and H2. Host H1 contains 3 virtual machines; VM1, VM2 and VM3. Host H2 contains two virtual machines; VM4 and VM5.

TABLE II
System Specifications

Node Specifications	✓ Intel (R) Xeon (R) Processor E5 - 2695 v4 2.10 GHz ✓ 45 M cache ✓ 64 GB DDR4 RAM ✓ 4 TB Hard Disk ✓ dual NIC card, ✓ Each processor-2 sockets ✓ Each socket-18 cores ✓ Total-36 logical processors
Virtual machine specifications	✓ 13 vCPU ✓ 16 GB DDR4 RAM ✓ 1 TB Hard disk.

Table II describes the system specifications. VM1 processes the user requests, VM2 handles the application logic and a database runs on VM3. A work-loader and a program to inject anomalies run on VM4 and VM5 respectively. The work-loader generates service for 24 hours on H1 in which the auction site is residing. For evaluating our proposed work, we continuously audit the CPU utilization data which we have named as Dataset 1. Initially the framework builds the base model from clear benign data. After that it diagnoses anomalies, if the CPU utilization varies significantly from the base model. The proposed framework not only detects anomalies but also automatically rebuild the model, if the fundamental data pattern changes.

For the next scenario, we have experimented with the proposed framework based on the Yahoo Cloud Serving Benchmark (YCSB) [8] framework to generate an expandable workload. YCSB has been run with varying workloads on different VMs. The framework regularly collects the data stream which we have called as Dataset 2. Here the database

system is a MySQL database. In the load period, data are loaded in the YCSB run. A number of transactions are used on the database to increase the load abnormally; hence generating anomalous data stream. Several performance metrics such as CPU utilization, memory utilization etc. are generated. Likewise in scenario 1, here also we are taking the CPU utilization data only. For both the scenarios, the time window is fixed experimentally to get optimal results. Similar to Dataset1, here also we build a base model from clear benign data. The streaming data is compared with the base model to detect abnormal CPU utilization. Moreover the framework automatically rebuilds the base models when there is a significant change in the fundamental data pattern. After that it diagnoses anomalies, if the CPU utilization varies significantly from the base model.

B. Results

In this experiment, we detect anomalies on the fly. Data coming in a streaming fashion is collected in a predefined time window. The algorithm raises an alarm when it detects anomalies in a window. For both the datasets we take 20 bins and the number of data points in a time window varies in the range of 98 - 110 and computed the threshold using R. For Dataset 1 we are considering a time window of 2 minutes duration, thus 30 time windows per hour and hence 720 windows (30*24) per day. Out of 720 windows, the anomaly injector injects anomalies to 120 windows. Thus we have 600 benign windows and 120 anomalous windows. The virtual machine and host metrics are collected/analysed in an anomaly detector. The proposed Algorithm 3 detects variations in the performance metrics of virtual machines and hosts. It correctly classifies 596 benign windows out of 600 and remaining 4 benign windows are misclassified. It also identifies 118 anomalous windows out of 120 and remaining 2 windows are misclassified. When a new window arrives, the algorithm compares its distribution with the model. If it matches, it classifies the window as benign; otherwise as anomalous and an alarm is raised.

In second scenario, for Dataset 2 we are considering a time window of 2 minutes duration, and we are collecting 2000 windows out of which 1700 are benign and remaining 300 are anomalous. The proposed algorithm correctly classifies 1685 benign windows out of 1700 and remaining 15 benign windows are misclassified. It also identifies 298 anomalous windows out of 300 and remaining 2 windows are misclassified. The accuracy results of the proposed algorithm are based on the following statistical metrics.

$$TPR = \frac{\#\ of\ successful\ anomalous\ windows\ classified}{\#\ of\ total\ anomalous\ windows} \quad (3)$$

$$FPR = \frac{\#\ of\ unsuccessful\ benign\ windows\ classified}{\#\ of\ total\ benign\ windows} \quad (4)$$

$$\text{TNR} = \frac{\text{\# of successful benign windows classified}}{\text{\# of total benign windows}} \quad (5)$$

$$\text{FNR} = \frac{\text{\# of unsuccessful anomalous windows classified}}{\text{\# of total anomalous windows}} \quad (6)$$

TABLE III
PERFORMANCE ANALYSIS OF PROPOSED METHOD FOR DATASET 1

Method	Metrics			
	TPR	TNR	FNR	FPR
Proposed Method	98.33	99.33	1.67	0.67
Parzen Window Estimators	96.53	96.63	3.47	3.37
Negative Selection	95.75	93.03	4.25	6.97

TABLE IV
PERFORMANCE ANALYSIS OF PROPOSED METHOD FOR DATASET 2

Method	Metrics			
	TPR	TNR	FNR	FPR
Proposed Method	99.33	99.12	0.67	0.88
Parzen Window Estimators	97.47	97.43	2.53	2.57
Negative Selection	95.94	96.01	4.06	3.99

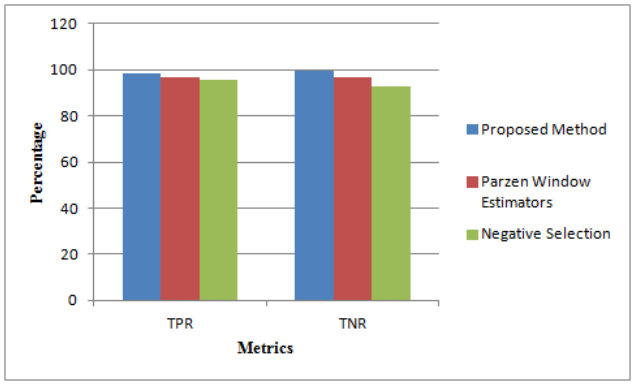

Fig. 2. Performance analysis of various schemes based on TPR and TNR for Dataset 1.

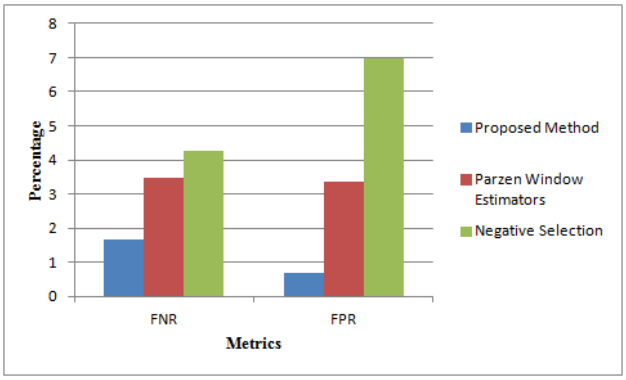

Fig. 3. Performance analysis of various schemes based on FNR and FPR for Dataset 1.

Tables III and IV reflect the overall accuracy statistics of the proposed method when applied in Dataset 1 and Dataset 2 respectively. Thus it shows the higher accuracy of our

proposed framework. Fig. 2 and Fig. 3 depicts the performance of the algorithm in terms of specified statistical metrics for Dataset 1. Fig. 4 and Fig. 5 depicts the performance of the algorithm in terms of specified statistical metrics for Dataset 2. Thus it shows the higher accuracy of our proposed method.

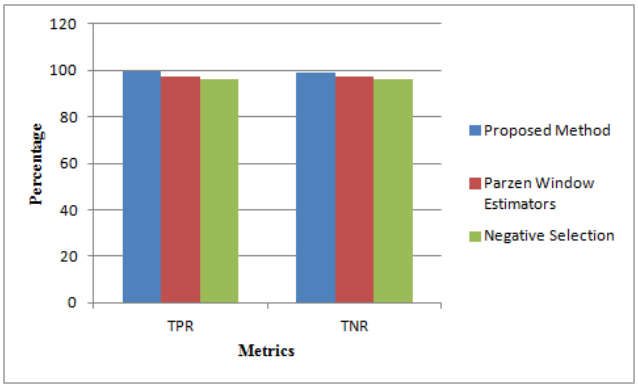

Fig. 4. Performance analysis of various schemes based on TPR and TNR for Dataset 2.

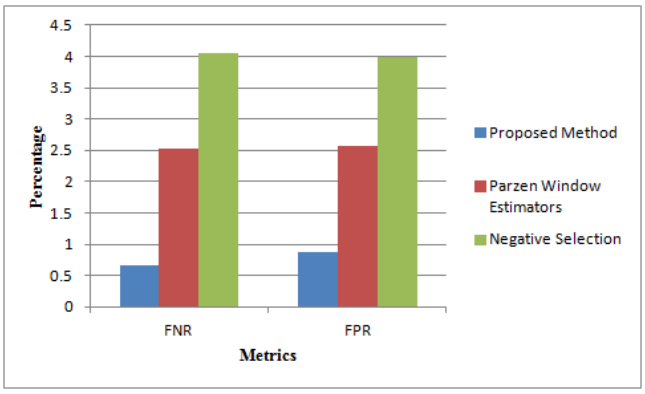

Fig. 5. Performance analysis of various schemes based on FNR and FPR for Dataset 2.

VI. Conclusions

Failure to detect anomalies/unusual characteristics will lead to massive negative impact in profit on account of inadequacy of responsiveness and availability in cloud data centers. Thus there is a need to improve cloud data center performance by establishing frameworks for detecting anomalies in real time. In this paper, an online anomaly detection framework using non-parametric statistical technique for cloud data centers is proposed. The proposed framework is capable of self-regulating the base model, when there is significant variation in the fundamental pattern of data. The experimental results show the higher accuracy of our proposed framework.

Acknowledgment

The research work reported in this paper is supported by Department of Electronics & Information Technology (DeitY), a division of Ministry of Communications and IT, Government of India, under Visvesvaraya PhD scheme for Electronics & IT.

References

[1] B. Li, B. Li and F. Liu, "Cloud and data center performance [Guest Editorial]," IEEE Network, 27, (2013): 6-7. doi: 10.1109/MNET.2013.6574658

[2] M. H. Ghahramani, M. Zhou, and C. T. Hon, "Toward cloud computing QoS architecture: Analysis of cloud systems and cloud services." IEEE/CAA Journal of Automatica Sinica 4.1 (2017): 6-18. doi: 10.1109/JAS.2017.7510313

[3] Smrithy G S, Sathyan Munirathinam, and Ramadoss Balakrishnan . "Online Anomaly Detection using Non-Parametric Technique for Big Data Streams in Cloud Collaborative Environment ." Big Data (Big Data), 2016 IEEE International Conference on, IEEE, 2016:1950-1955. doi: 10.1109/BigData.2016.7840816

[4] Chandola, Varun, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey." ACM computing surveys (CSUR) 41.3 (2009): 15. doi: 10.1145/1541880.1541882

[5] S. Shen, V. V. Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters,"15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (2015): 465–474. doi: 10.1109/CCGrid.2015.60

[6] S. Barbhuiya, Z. Papazachos, P. Kilpatrick and D. Nikolopoulos, "RADS: Real-time Anomaly Detection System for Cloud Data Centres." arXiv preprint arXiv:1811.04481 (2018).

[7] RUBiS testbed.[Online], 2003. Available: http://rubis.ow2.org/

[8] Cooper, Brian F., et al. "Benchmarking cloud serving systems with YCSB." Proceedings of the 1st ACM symposium on Cloud computing. ACM, 2010. doi: 10.1145/1807128.1807152

[9] Ibidunmoye, Olumuyiwa, Francisco Hernández-Rodriguez, and Erik Elmroth. "Performance anomaly detection and bottleneck identification." ACM Computing Surveys (CSUR) 48.1 (2015): 4. doi: 10.1145/2791120

[10] Krügel, Christopher, Thomas Toth, and Engin Kirda. "Service specific anomaly detection for network intrusion detection." Proceedings of the 2002 ACM symposium on Applied computing. ACM, 2002. doi: 10.1145/508791.508835

[11] Kruegel, Christopher, et al. "Bayesian event classification for intrusion detection." Computer Security Applications Conference, 2003. Proceedings. 19th Annual. IEEE, 2003.doi: 10.1109/CSAC.2003.1254306

[12] Parzen, Emanuel. "On estimation of a probability density function and mode." The annals of mathematical statistics 33.3 (1962): 1065-1076. doi:10.1214/aoms/1177704472

[13] Yeung, Dit-Yan, and Calvin Chow. "Parzen-window network intrusion detectors." Pattern Recognition, 2002. Proceedings. 16th International Conference on. Vol. 4. IEEE, 2002.doi: 10.1109/ICPR.2002.1047476

[14] Song, Xiuyao, et al. "Conditional anomaly detection." IEEE Transactions on Knowledge and Data Engineering 19.5 (2007). doi: 10.1109/TKDE.2007.1009

[15] Clifton, David A., et al. "An extreme function theory for novelty detection." IEEE Journal of Selected Topics in Signal Processing 7.1 (2013): 28-37. doi: 10.1109/JSTSP.2012.2234081

[16] Hoare, Stephen W., David Asbridge, and Paul CW Beatty. "On-line novelty detection for artefact identification in automatic anaesthesia record keeping." Medical engineering & physics 24.10 (2002): 673-681. doi: 10.1016/S1350-4533(02)00146-7

[17] Qiao, Yan, et al. "Anomaly intrusion detection method based on HMM." Electronics letters 38.13 (2002): 663-664. doi: 10.1049/el:20020467

[18] Sun, Bo, et al. "Anomaly detection based secure in-network aggregation for wireless sensor networks." IEEE Systems Journal 7.1 (2013): 13-25. doi: 10.1109/JSYST.2012.2223531

[19] Yeung, Dit-Yan, and Calvin Chow. "Parzen-window network intrusion detectors." Pattern Recognition, 2002. Proceedings. 16th International Conference on. Vol. 4. IEEE, 2002. doi: 10.1109/ICPR.2002.1047476

[20] González, Fabio A., and Dipankar Dasgupta. "Anomaly detection using real-valued negative selection." Genetic Programming and Evolvable Machines 4.4 (2003): 383-403. doi: 10.1023/A:1026195112518

[21] Cabrera, João BD, Lundy Lewis, and Raman K. Mehra. "Detection and classification of intrusions and faults using sequences of system calls." Acm sigmod record 30.4 (2001): 25-34. doi: 10.1145/604264.604269

[22] Ghosh, Anup K., Aaron Schwartzbard, and Michael Schatz. "Learning Program Behavior Profiles for Intrusion Detection." Workshop on Intrusion Detection and Network Monitoring. Vol. 51462. 1999.

[23] Farid, Dewan Md, Nouria Harbi, and Mohammad Zahidur Rahman. "Combining naive bayes and decision tree for adaptive intrusion detection." arXiv preprint arXiv: 1005.4496 (2010).
[24] Hodge, Victoria J., and Jim Austin. "A survey of outlier detection methodologies." Artificial intelligence review 22.2 (2004): 85-126. doi: 10.1007/s10462-004-4304-y
[25] Patcha, Animesh, and Jung-Min Park. "An overview of anomaly detection techniques: Existing solutions and latest technological trends." Computer networks 51.12 (2007): 3448-3470. doi: 10.1016/j.comnet.2007.02.001
[26] Buzen, Jeffrey P., and Annie W. Shum. "Masf-multivariate adaptive statistical filtering." Int. CMG Conference. 1995.
[27] Datar, Mayur, et al. "Maintaining stream statistics over sliding windows." SIAM journal on computing 31.6 (2002): 1794-1813.
[28] Aron, Arthur, and Elaine N. Aron. Statistics for psychology. Prentice Hall/Pearson Education, 2003.
[29] Abouzakhar, Nasser, and Abu Bakar. "A Chi-square testing-based intrusion detection Model." Procs 4th International Conference on Cybercrime Forensics Education & Training. 2010.
[30] W.H.Press. Numerical recipes in c. [Online], 1986. Available: http://www.aip.de./groups/soe/local/numbers/book pdf/c14-3.pdf
[31] Wang, Chengwei, et al. "Statistical techniques for online anomaly detection in data centers." Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on. IEEE, 2011. doi: 10.1109/INM.2011.5990537
[32] R. The r project for statistical computing. [Online], 2015. Available: http://www.r-project.org/
[33] SAS analytics software and solutions. [Online], 2013. Available: http://www.sas.com/en_in/home.html

Smrithy G S received the B.Tech degree in Information Technology from University of Kerala in 2011 and M.Tech degree in Engineering Statistics (University topper) from Cochin University of Science and Technology in 2014. Currently, she is a PhD researcher in the Department of Computer Applications at National Institute of Technology, Tiruchirappalli, India. Her research interests include: Statistical techniques for security and privacy in Big data, Anomaly detection and protection, identity management and intrusion detection techniques in Cloud Computing, and Information Security.

Ramadoss Balakrishnan received the M.Tech degree in Computer science and Engineering in 1995 from the Indian Institute of Technology, Delhi and the PhD degree in Applied Mathematics in 1983 from Indian Institute of Technology, Bombay. Currently he is working as a Professor of Computer Applications at National Institute of Technology, Tiruchirapalli. His research interests include: Software Testing Methodologies, Security and Privacy in Big Data and Cloud, software Metrics, Data Warehouse – EAI, Data Mining, WBL, and XML. He is a recipient of Best Teacher Award at National Institute of Technology, Tiruchirapalli, India during 2006-2007. He is a member of IEEE, Life Member (LM) of ISTE, New Delhi, Life Member (LM), Computer Society of India.