

RAIN: Towards Real-Time Core Devices Anomaly Detection Through Session Data in Cloud Network

Haoyu Liu^{*}, Chongrong Fang^{*}, Yining Qi^{*}, Jian Bai[†], Shaozhe Wang[†], Xiong Xiao[†],
Daxiang Kang[†], Biao Lyu[†], Peng Cheng^{*}, Jiming Chen^{*}

^{*}Zhejiang University, Hangzhou, China

[†]Alibaba Group, Hangzhou, China

Email: {haoyu_liu, qyning710, cjm}@zju.edu.cn, chongrongfang.zju@gmail.com, pcheng@iipc.zju.edu.cn
{jian.baij, shaozhe.wsz, xiong.xiaox, daxiang.kdx, lubiao.lb}@alibaba-inc.com

Abstract—Core devices form the critical components of the cloud network and provide service to multiple tenants simultaneously. The anomalies that happened in core devices impact network availability of a large number of users, meanwhile, lead to the degradation of cloud providers' profits. However, direct monitoring of core devices needs to deploy massive heartbeat checking tools on numerous related components, which will be extremely laborious. In this paper, we deploy RAIN to reduce the number of devices that need to be detailed investigated for anomalies. The session traffic data among core devices and served virtual machines are utilized to conduct the analyzing. To guarantee near real-time monitoring, RAIN is designed as a two-step structure and incorporating four feature-based detection methods. RAIN has been deployed in Alibaba's production cloud network for over 6 months and is analyzing terabytes of traffic flow metrics per day.

Index Terms—Anomaly detection, cloud network, quality of service

I. INTRODUCTION

In recent years, we have witnessed a widespread trend to adopt cloud services from cloud vendors such as Amazon, Microsoft, Google Cloud, and Alibaba Cloud. Unlike traditional computation procedure which is locally executed, cloud computing provides remote computation and storage resources for users to “rent”. With multi-tenant architecture, the infrastructure of the cloud system is usually shared by thousands of different users. This benefits tenants in lower cost and higher convenience, however, can increase the extent of the negative influence when the system occurs the anomalies.

Till now, numerous unexpected incidents have happened in the commercial cloud system. In August 2013, a downtime of Amazon happened due to system faults, resulted in a loss of about \$5 million [1]. In June 2018, a network error happened in Alibaba Cloud and caused a half-hour transaction data loss, which influenced thousands of e-commerce business. In March 2019, a breakdown of Alibaba Cloud Elastic Container Service in North China-2 region led to a massive service outage of websites and apps in north China. While the amount of cloud-backed services is growing, such incidents will cause much

more economic loss. Therefore, from the perspective of both cloud providers and users, it is of significant importance to detect cloud anomalies and recover service from outage timely.

However, compared with the traditional network, the extremely large number of software/hardware devices, the complex network structure, the multi-tenant property, and the continuous dynamic change of the cloud network result in a big gap to use traditional monitoring methods on the cloud system. For example, heartbeat checking is widely adopted in the traditional network system, however, in the cloud, virtualized devices are updated frequently, thus the heartbeat checking is prone to generate false alarms. Besides, there are much more devices in the cloud network, thus it will be laborious and expensive to implement heartbeat checking to cover all of the concerned components.

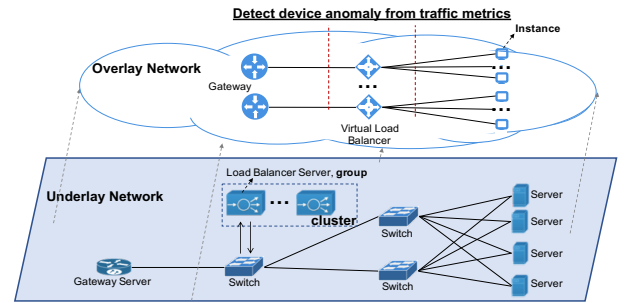


Fig. 1: An example topology of load balancer.

Therefore, in this paper we propose and deploy a method to pick out most suspicious core devices, such as core switches and load balancers, for alleviating the burden of utilizing traditional monitoring methods in the cloud-scale system. In the production cloud network, millions of different kinds of core devices form the foundation of the cloud system and support packet transmissions among tenants and the internet. Moreover, core devices are generally virtualized and installed on dedicated servers [2], [3], and each core device is composed of numerous sub-components. Example topology of a common core device, the load balance device, is demonstrated in Fig. 1. To support its normal operation status, massive sub-components from both underlay and overlay networks

This work was supported by Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies.

are deployed. To justify whether a load balance device is in abnormal, a traditional idea is to implement methods such as heartbeat checking on every component. However in this paper, rather than directly monitoring all of these sub-components, we introduce a pre-filtering mechanism based on the in/out session metrics of the core device to pick out most suspicious devices. In Fig. 1, the session metrics collected from the software of the virtualized load balance device, can reflect the overall running status of the the core device. Without loss of generality, we define this idea that utilizes session metrics to estimate the status of core devices as the core device anomaly detection.

However, there are several challenges to explore session data and conduct real-time detection. First, the data volume is extremely huge. Thus we design a two-step structure of the detection mechanism and only conduct first-step online. Then, the anomalies are coming in different shapes. Thus we propose several feature-based methods to quantify the normal status of session data and pick devices with unmatched session data as suspects. Third, in practical scenarios, the interpretability of the output result is desired for network operators' fast decisions. Thus the designed methods are based on several intuitive features of session traffic.

We summarize our contributions as follows:

- To the best of our knowledge, this is the first work that tries to pick suspicious core devices from cloud network based on session data.
- A two-step detection structure incorporating four intuitive feature-based methods is proposed.
- We conduct evaluations on a real-world historical data set. The designed system has been deployed on Alibaba's production cloud network for over 6 months. The results produced by RAIN are heavily used by network operators and developers to monitor and analyze suspected anomalies.

The rest of this paper is organized as follows. We discuss related works in Sec. II. We demonstrate background and system overview in Sec. III. Four feature-based anomaly detection methods for traffic flow metrics are presented in Sec. IV. We provide the evaluation results in Sec. V and draw the conclusion in Sec. VI.

II. RELATED WORKS

It has been a heated topic for both academia and industries to investigate anomaly detection for the network. However, many previous works are not quite suited for the cloud network. [4] designed an anomaly detection method for network devices through server-generated log data. Due to the lack of labels, a data-driven framework combining a word representation method and positive-unlabeled learning model is proposed. [5] tried to predict hardware abnormal events of switches by extracting information from system log data. [6] designed a distribution-based anomaly detection method for monitoring cloud performance, using CPU utilization and system-related KPIs. However, [4]–[6] ignored the fact that

the cloud system is updating continuously. When the virtualized core devices are updating, the server-generated log data and CPU utilization data will be changed significantly and influence the detection results.

Several recent works detect network anomalies from end hosts. In [7], a network-wide packet monitoring system Trumpet was designed and implemented. The system provided an active event description method and a packet match algorithm at the end host to detect anomalies in network applications. In [8], authors deployed a TCP metrics analysis system on client machines to distinguish the root cause of network failure between customer's and provider's systems. However, due to the private and privacy policy, most cloud network providers can not have access to the end host virtual machines, thus these works are inappropriate for the cloud service provider.

In addition, some previous works propose different kinds of network anomalies detection methods using session data. Such as in [9], authors developed a PCA based method to separate the network traffic into normal and anomalous components. In [10], authors designed a method that quantified the simultaneous increase and decrease of a group of flows' traffic volumes to detect a new class of anomalies. In [11], authors investigated entropy-based anomaly detection methods through network traffic metrics to detect botnet-like malware. However, none of them have drawn the focus on the core devices.

III. BACKGROUND AND SYSTEM OVERVIEW

In the cloud network, core devices like core switches and load balance devices are responsible for forwarding customers' packets from the backbone network to individual virtual machines (VM). An example topology is presented in Fig. 1. Usually, one core device serves hundreds to thousands of VMs. Therefore, once there is a fault happened inside a core device, a large number of tenants will be influenced. We define each VM as an *instance*, the set of VMs which are served by the same core device as a *group* of instances, the group of core devices as a *cluster* of groups. Taking advantage of the software-defined network, we directly collect session data from the installed software in each core device. The fields of collected raw data are listed in Table I. Note that the raw data is in instance-level, therefore an aggregation function is deployed on each device to obtain group-level data, which reflects the session metrics of the corresponding device.

TABLE I: Raw data fields.

Field name	Description
instance ID	A unique identifier of VMs.
connsPS	The value of currently established links per second.
failConnPS	The value of currently failed links per second.
inBitsPS	Inflow volume per second.
outBitsPS	Outflow volume per second.
msTimestamp	Timestamp for the data.

For a production cloud network, the more timely detection and recovery of the anomalies can lead to a smaller economic loss. To satisfy the requirement of timeliness and prevent influencing tenants' network performances, we deploy RAIN through a bypass computing system. The system is supported by JStorm [12], which is a widely used stream process engine inside Alibaba. However, due to the significant large scale of cloud tenants and session metrics (millions of tenants are currently using cloud services from Alibaba), the computation workload is massive. Therefore, we design a two-step structure to fulfill computation requirements. We demonstrate the system structure in Fig. 2. First, RAIN Customs is operating as a coarse-grained detector and is deployed to monitor every target session data in real-time. The object of RAIN Customs is to filter out core devices that are least suspicious but pick out the most suspicious for the following detailed detection, which will then be conducted in Interviewing Room. As the

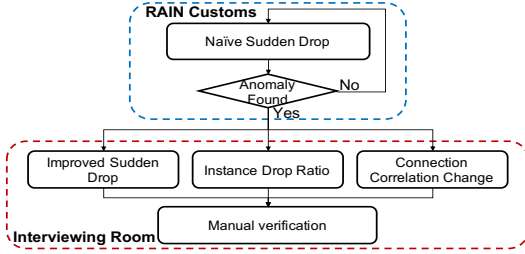


Fig. 2: Detection pipeline structure.

second step, Interviewing Room is not continuously running in the detection system, but can be triggered when there is suspicious core devices reported by RAIN Customs. The output of Interviewing Room is the intersection of the outputs from three detection methods. Final result will be transmitted to network administrators, who will then utilize traditional detection tools such as heartbeat checking for each reported core device, to verify and diagnose the anomalies.

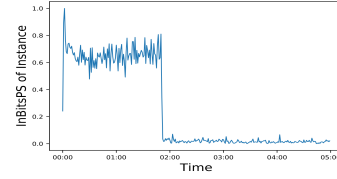
IV. DETECTION METHODS

In this section, we introduce four feature-based detection methods used for our two-step core devices anomaly detection system RAIN. We design these feature-based methods by quantifying several intuitive features, such as the sudden change of session traffic volume and irregular connection status. The methods are assigned for RAIN Customs and Interviewing Room respectively.

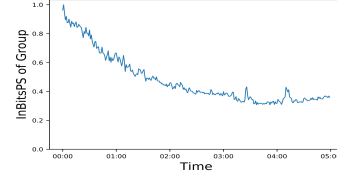
A. RAIN Customs Method

RAIN Customs will be continuously running in the detection system, thus desires easily and efficient methods to filter out the least suspicious devices. We design a method based on session traffic volume. In the cloud network, a core device usually serves thousands of tenants' instances. A single instance's traffic volume has scarcely impact on device-level(group-level) traffic volume. For example, Fig.

3(a)¹ demonstrates the session traffic of a single instance. At about 1:50, there was a shutdown operation of the client's virtual machine, which lead to a sudden drop in the session flow. Fig. 3(b) demonstrates the session traffic of the group



(a) Flow volume of a single instance.



(b) Flow volume of the corresponding group.

Fig. 3: Traffic flow volume examples.

which contains the instance of Fig. 3(a). Although there were changes happened in single users' session traffic, the overall traffic volume was still stable. Two factors can cause an obvious change in core devices' session traffic. First, a traffic-related change on the core devices, such as traffic shifting, which are usually conducted by network operators and automatically logged by the system. Second, a fault happens in the core device, which is unexpected and can lead to packet loss among core devices and served instances. Therefore, we deploy a coarse-grained detection method to monitor group-level session traffic and pick out devices that have abnormal sudden change while not logged with traffic-related operations. In this paper, we investigate the irregular downwards trend of session traffic. And all of the proposed methods can be easily conducted in the upwards trend scenario. For the RAIN Customs, we propose a naive sudden drop detection method (NSD) based on z-score to conduct online monitoring. The NSD value can be obtained by:

$$z = \frac{x - \mu}{\sigma} \quad (1)$$

where x is the traffic volume of current aggregated group-level session traffic (aggregated inBitsPS/outBitsPS), μ and σ are the mean and standard deviation of the sliding window respectively. We preset 10 consecutive values as the length of the sliding window. Moreover, a threshold Z_{thre} is assigned to determine whether there is an anomaly. When the NSD value is lower than Z_{thre} , a suspicious device is reported.

B. Interviewing Room Methods

Interviewing Room will be "triggered" when there are suspicious devices reported by the previous NSD method. Three

¹In this paper, all absolute data values in our examples have been transformed to preserve anonymity.

other feature-based methods are designed in Interviewing Room for a further investigation of the suspected anomalies.

1) *Improved Sudden Drop (ISD)*: NSD used in the RAIN Customs is a coarse-grained monitor investigating group-level changes. Some special scenarios will degrade the performance of NSD and induce false positives. Commonly, the instances in a group belong to different tenants. When there is a single tenant's traffic flow dominating the corresponding group's session traffic, the device-level NSD value can be significantly influenced by this user's behaviors and generate false alarms. To eliminate such kinds of false alarms, we propose an improved method, ISD, to analyze instance-level flow volume and obtain a multi-instance behavior indicator for anomaly detection. The method can distinguish alarms caused by several dominating tenants through the following model:

$$\begin{aligned} \min \quad & drop(k) = m \\ \text{s.t.} \quad & \sum_{i=1}^m \frac{drop_i}{drop_{all}} \geq 80\%, \quad drop_{all} = \sum_{i=1}^M drop_i \end{aligned}$$

k is the timestamp, $drop_i$ is the drop value of instance i , M is the total number of instances in the group, $drop_{all}$ denotes the overall decline of the group. The model can obtain the minimum number of instances, whose sum of traffic decline accounts for over 80% of the group-level session traffic decline. Therefore, the lower the m indicates the greater dominance of the dominant instances and implies a higher possibility of a false positive. The algorithm to obtain ISD

Algorithm 1 Improved Sudden Drop Indicator

- 1: **Input:** $drop_i$ for $i = 1, 2, \dots, M$
 - 2: **Output:** $drop(k)$ where k is current timestamp
 - 3: $drop_{all} \leftarrow$ summation of $drop_i$
 - 4: $drop_{ib2s} \leftarrow$ rank $drop_i$ in descending order
 - 5: $temp_{sum} \leftarrow 0$
 - 6: $m = 0$
 - 7: **while** $\frac{temp_{sum}}{drop_{all}} < 80\%$ **do**
 - 8: $temp_{sum} \leftarrow temp_{sum} + drop_{ib2s}[m]$
 - 9: $m \leftarrow m + 1$
 - 10: **end while**
 - 11: $drop(k) \leftarrow m$
-

value is demonstrated in Algorithm 1. To obtain $drop(k)$, we first rank all $drop_i$ and calculate the accumulated value from large to small until the summation exceeds 80% of the total decline value, then count the number to get m and $drop(k)$.

To determine whether the reported anomaly is a false alarm, we compare the calculated $drop(k)$ with the average amount of $drop(k)$ in the last normal day:

$$z(k) = drop(k) - m_{avg} \quad (2)$$

m_{avg} is the average $drop(k)$ of the previous normal day. In Fig. 4, we present an example frequency histogram for $z(k)$ in 1 hour. According to its distribution, we develop a cusum-based method to monitor the change of $z(k)$:

$$S(k) = \max(0, (S(k-1) + z(k))), \quad S(0) = 0 \quad (3)$$

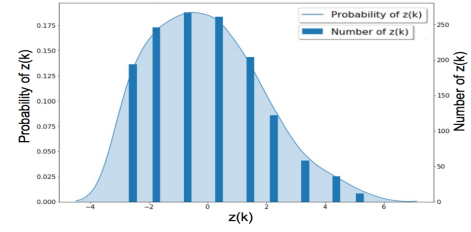


Fig. 4: Frequency histogram of $z(k)$.

To determine the anomaly, a threshold C_{thre} is used to compare with the calculated result $S(k)$:

$$C_{thre} = \alpha S_{max} \quad (4)$$

where α is an adjustable coefficient and S_{max} denotes the biggest $S(k)$ in the previous normal day.

2) *Instance Drop Ratio (IDR)*: Another feature-based method is to quantify the ratio of the decreasing instances. Commonly, due to the instances within one group belong to different tenants, the upward or downward trend of every instances' traffic should not be the same. Therefore, when a large portion of instances' traffic decreasing or increasing together, the root cause may blame the related core devices rather than VMs. IDR indicator is designed to distinguish this scenario. To implement IDR, we first filter out instances have frequent small fluctuation in session traffic by calculating 1-step lag Pearson correlation coefficient. Next, we mark other instances as upwards or downwards to indicate their trend. Taking downward trend as an example, we count the number of decreasing instances and calculate percentage by:

$$P = \frac{num_a}{num} \quad (5)$$

num_a is the number of downward instances and num is the total number of instances in the group. To determine the anomaly, a threshold P_{thre} is proposed for comparison. The significantly large of IDR value indicates an anomaly.

3) *Connection Correlation Change (CCC)*: Besides monitoring session traffic volume, we further develop a method based on the connection status. In the cloud network, it is common that a small percentage of links fail to establish due to the non-exist destination IP, forwarding limitation, and etc. Using the device-level aggregated session metrics, the correlation between the currently established links (connsPS) and the failed links (failConnPS) of a group of instances is usually stable.

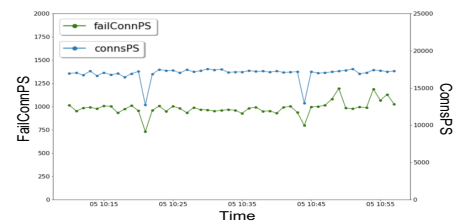


Fig. 5: Comparison between connsPS and failConnPS.

An example is shown in Fig. 5. The trends of two indexes are roughly the same. We develop CCC method to quantify the relation between connsPS and failConnPS. A euclidean distance based method is implemented as follows:

$$d_t = \sqrt{(x_{t-9} - y_{t-9})^2 + \dots + (x_t - y_t)^2} \quad (6)$$

$x_{t-9}, \dots, x_t$ and $y_{t-9}, \dots, y_t$ denote 10 consecutive values of connsPS and failConnPS till timestamp t , respectively. To spot the change, we calculate an absolute difference between two consecutive timestamps by:

$$D_{correlation} = |d(x, y)_t - d(x, y)_{t-1}| \quad (7)$$

A threshold $F_{thre} = \beta\sigma$ is assigned to determine the anomaly. σ is the deviation of d_t values in the last normal day.

V. PERFORMANCE EVALUATION

In this section, we evaluate the proposed methods on the historical data collected from our production cloud network, furthermore, present the deployment experience. We collect session data metrics of 6 months from a production core device as the historical data set. There are 2880 instances in this group. The total number of data points is about 4.5×10^{10} .

A. Detection Methods Evaluation

We conduct multiple experiments to evaluate the proposed methods. It is worth noting that anomalies are relatively few in the production system. Meanwhile, due to anomalies' serious threat to the profits, cloud providers prefer finding as many abnormal devices as possible. Thus, the recall rate is of significant importance for the detection results.

There are several coefficients in the designed detection system. In Fig. 6, we demonstrate various precision and recall rates using historical data, based on different threshold values. The left and right axes denote precision rate and recall rate respectively. We select the coefficients' value from the overlaps area of two curves. Although the best selection may

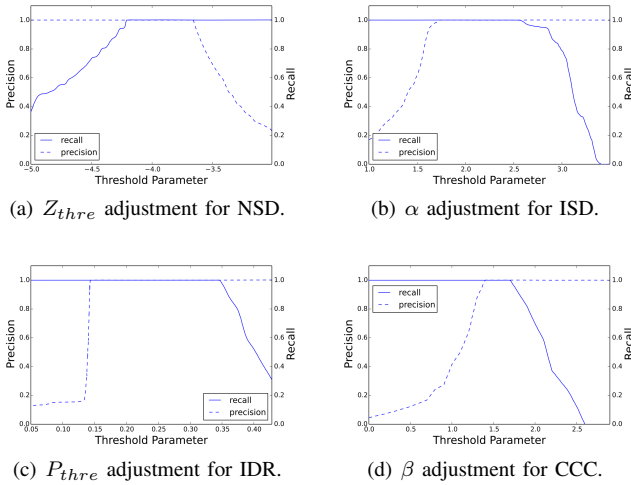


Fig. 6: Coefficients adjustment examples.

not exist, we can have a preferred selection based on the

priority of recall rate. Thus we first guarantee the highest recall rate when there is not any selection can satisfy both recall and precision at the same time. In the following evaluation, the coefficients are configured as: $Z_{thre} = -4$, $\alpha = 2$, $P_{thre} = 0.2$, $\beta = 1.5$.

For each module, we calculate corresponding recall rate, precision and F1-score by:

$$\text{recall} = \frac{TP}{TP + FN} \quad (8)$$

$$\text{precision} = \frac{TP}{TP + FP} \quad (9)$$

$$F1_{\text{score}} = \frac{2TP}{2TP + FN + FP} \quad (10)$$

TP denotes true positive, TN denotes true negative, FP denotes false positive, FN denotes false negative, Performances are shown in Table II. The results indicate that our main object

TABLE II: Detection performance on historical data

Performance indicator	RAIN Customs	Interviewing Room
Recall rate	100%	50%
Precision	4.7%	33.3%
F1-score	0.09	0.46

which is to maintain a high recall rate is guaranteed by RAIN Customs. All of true abnormal devices are picked out by the first-step of detection. Then, Interviewing Room has a largely improved precision rate, which reaches 33.3%. Therefore, suspected anomalies selected by Interviewing Room have a higher probability to be the true positives compared to the NSD method. Thus, operators first verify and recover the detected anomalies from Interviewing Room. If the abnormal status is not fixed, operators will then investigate other output from RAIN Customs.

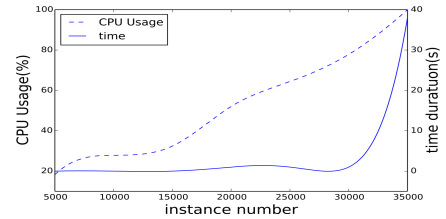


Fig. 7: System operation efficiency evaluation.

The processing speed of the continuously running step, RAIN Customs, is demonstrate in Fig. 7. The processing duration is less than 5 seconds and grows slowly before the number of instances reaches 30k. Meanwhile, the majority of core devices serve less than 30k instances in our production cloud network. Thereby, the result indicates a satisfactory running speed.

B. Case Study

We provide a case study to demonstrate the operation process and detection results of RAIN. First, RAIN Customs conducts preliminary evaluation and sort out suspected anomalies in real-time. Then, if there is any suspected anomaly, the

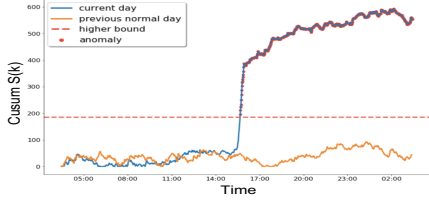


Fig. 8: Improved sudden drop detection method.

corresponding session data will be transmitted to Interviewing Room for further evaluation. For ISD, an example is demonstrated in Fig. 8. In normal status, ISD value evolve near 0 value. When there are anomalies happened, the value of $S(k)$

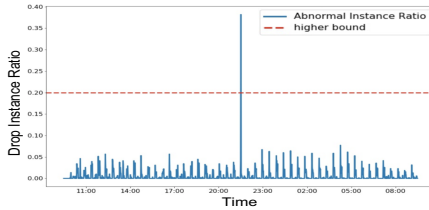


Fig. 9: Instance drop ratio detection method.

rapidly increases cross the threshold. For IDR, an example is demonstrated in Fig. 9. The significant increase of IDR value indicates a high possibility of devices' anomaly. For CCC, an example is demonstrated in Fig. 10. Euclidean distance is used to quantify the correlation between two connection performance indicators. When the CCC value is over the threshold, the related devices will be treated as suspicious devices. These three methods are conducted equally. Results

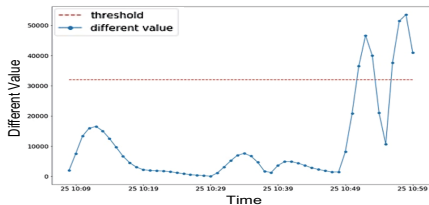


Fig. 10: Connection correlation change detection method.

are intersected as the final output. Due to the higher precision rate, network administrators will then first verify the detected suspicious devices from Interviewing Room. If the abnormal status cannot fix, other suspects detected by RAIN Customs will be further investigated.

VI. CONCLUSION

Considering the laborious and expensive efforts to implement traditional network monitoring methods such as heart-

beat checking on the cloud network, in this paper, we have presented a “pre-filter” system for alleviating the monitoring burden by distinguishing most suspicious core devices and filtering out normal devices. To fulfill the desire for timely detection and the processing of massive related data metrics, we proposed a two-step structure of the detection system RAIN. The first-step of RAIN, the RAIN Customs, is designed to conduct online filtering and report all true anomalies. Experiments on historical data indicate 100% recall rate of RAIN Customs. The second-step Interviewing Room is designed to pick out higher priority suspicious devices. Experiments show that the precision rate is growing a lot compared with RAIN Customs, thus can provide higher suspicious core devices for higher priority in the followed detailed verification process. We have deployed the system in the real-world running production cloud network for over 6 months. The system is currently analyzing terabytes of session traffic metrics per day and leasing the burden for monitoring core devices' anomalies.

REFERENCES

- [1] 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 Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 1, pp. 61–75, 2016.
- [2] D. E. Eisenbud, C. Yi, C. Contavalli, C. Smith, R. Kononov, E. Mann-Hielscher, A. Cilingiroglu, B. Cheyney, W. Shang, and J. D. Hosein, “Maglev: A fast and reliable software network load balancer,” in *13th Symposium on Networked Systems Design and Implementation (NSDI)*, 2016, pp. 523–535.
- [3] M. Dalton, D. Schultz, J. Adriaens, A. Arefin, A. Gupta, B. Fahs, D. Rubinstein, E. C. Zermeno, E. Rubow, J. A. Docauer *et al.*, “An-dromeda: performance, isolation, and velocity at scale in cloud network virtualization,” in *15th Symposium on Networked Systems Design and Implementation (NSDI)*, 2018, pp. 373–387.
- [4] W. Meng, Y. Liu, S. Zhang, D. Pei, H. Dong, L. Song, and X. Luo, “Device-agnostic log anomaly classification with partial labels,” in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, 2018, pp. 1–6.
- [5] S. Zhang, Y. Liu, W. Meng, Z. Luo, J. Bu, S. Yang, P. Liang, D. Pei, J. Xu, Y. Zhang *et al.*, “Prefix: Switch failure prediction in datacenter networks,” *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 1, p. 2, 2018.
- [6] C. Wang, V. Talwar, K. Schwan, and P. Ranganathan, “Online detection of utility cloud anomalies using metric distributions,” in *2010 IEEE Network Operations and Management Symposium-NOMS 2010*, 2010, pp. 96–103.
- [7] M. Moshref, M. Yu, R. Govindan, and A. Vahdat, “Trumpet: Timely and precise triggers in data centers,” in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 129–143.
- [8] B. Arzani, S. Ciraci, B. T. Loo, A. Schuster, and G. Outhred, “Taking the blame game out of data centers operations with netpoirot,” in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 440–453.
- [9] A. Lakhina, M. Crovella, and C. Diot, “Diagnosing network-wide traffic anomalies,” in *ACM SIGCOMM computer communication review*, vol. 34, no. 4, 2004, pp. 219–230.
- [10] F. Silveira, C. Diot, N. Taft, and R. Govindan, “Astute: Detecting a different class of traffic anomalies,” *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 267–278, 2011.
- [11] P. Berezinski, B. Jasiul, and M. Szyrka, “An entropy-based network anomaly detection method,” *Entropy*, vol. 17, no. 4, pp. 2367–2408, 2015.
- [12] Alibaba, “Jstorm: a distributed and fault-tolerant real-time computation system.” 2014. [Online]. Available: <https://github.com/alibaba/jstorm/wiki>