CloudPin: A Root Cause Localization Framework of Shared Bandwidth Package Traffic Anomalies in Public Cloud Networks

Shize Zhang[‡*], Yunfeng Zhao[‡*], Jianyuan Lu[§†], Biao Lyu[§],
Shunmin Zhu[§], Zhiliang Wang[‡], Jiahai Yang[‡¶†], Lin He[‡], Jianping Wu[‡]
[‡]Tsinghua University, China
[§]Alibaba Group, China
[¶]Peng Cheng Laboratory, China

***Abstract*—Due to the sharing nature of public cloud, most of the cloud services use a sharing bandwidth package (sBwp) model to conduct inbound/outbound communication. The sBwp model allows users to purchase a sharing bandwidth for plenty of virtual machines instead of purchasing bandwidth for each virtual machine separately. The advantage of sBwp is that it can provide users with convenient configuration and lower economic cost. However, the sBwp model brings new challenges for operators to localize the root cause of traffic anomalies of a sharing bandwidth, especially for a globally distributed large-scale public cloud with millions of users. In this paper, we first formalize the sBwp problem on the cloud and propose *CloudPin*, a root cause localization framework for this problem. Our framework solves all the challenges by employing a multi-dimensional algorithm with three sub-models of prediction deviation, anomaly amplitude, and shape similarity, and an overall ranking algorithm. Evaluations on real-world data, from one of the world-renowned public cloud vendors, show that our algorithm precision reaches 97.8% for the top 1 of the ranking list, outperforming multiple baseline algorithms.**

***Index Terms*—shared bandwidth package, root cause localization, traffic anomaly, public cloud**

I. Introduction

With the continuous development of cloud technology, the public cloud providers provide a wide range of services for large-scale users, such as cloud computing, storage, and networking [1]–[3]. According to the latest statistics, the worldwide cloud market grew 32% to $39.9 billion in the fourth quarter of 2020 [4]. With the ever-increasing number of users and services, the operation and maintenance of public clouds are facing more challenges. One of the most common services in public cloud networks is sharing of resources to achieve convenient management and low cost. For example, one physical machine can be shared by multiple virtual machines (VMs) [5]. The same network device can be shared by many services or network functions [6]. For cloud users, the common sharing of resources service is the shared bandwidth package (sBwp) service.

sBwp is a user-friendly traffic management service. Specifically, users with a large number of VMs can choose to use sBwp service to purchase aggregate egress bandwidth for all VMs, rather than purchasing bandwidth for each VM individually. This service has several benefits for users. The first one is the low economic cost. Users can use shared bandwidth to avoid wasting bandwidth resources when certain VMs are idle. The second one is the ease of application service management. With the development of microservice technology, many application services are now using distributed microservice architecture [7]. As a result, users often need multiple VMs to work together to complete an application service. sBwp service allows users to directly purchase egress bandwidth for all VMs required for an application service.

However, sBwp services bring new issues to the operation and maintenance of public clouds. One of the most critical issues is the problem of locating the root cause of traffic anomalies. Specifically, when users find anomalies in sBwp traffic, they want to locate the specific VM(s) in sBwp associated with the anomaly. Since most users lack the relevant operational and maintenance skills, they expect the cloud provider to provide diagnostic tools.

For public cloud service providers, the root cause localization of traffic anomalies in sBwp services requires addressing the following four challenges.

- **Large scale**. The number of users using shared bandwidth package services in the public cloud is vast, with virtual machines deployed worldwide. Users deploy services covering a wide range, such as gaming, live streaming, websites, etc. How to locate the root causes of traffic anomaly in this large-scale, multi-type application scenarios is a core challenge.
- **Diversity of traffic anomalies**. Due to the different service applications deployed by users in the cloud network, traffic anomalies exhibit diversity, including persistent and transient occurrences in the time dimension, spikes and troughs in the amplitude dimension, and high jitter in the frequency dimension. Therefore, the algorithm needs to accommodate different types of anomalies.
- **Low overhead and real-time**. Since cloud service providers need to provide root cause localization services to a multitude of users, the localization algorithm is

[*]Work done while these authors were interns at Alibaba Group.
[†]Co-corresponding authors.

constrained in computational and storage resources. In addition, the cloud service provider needs to respond to user requests as quickly as possible.
- **Dynamicity**. A large number of users frequently change the deployment of virtual machines and network devices, benefiting from the powerful dynamic tuning capabilities of cloud networks. This requires the algorithm to adapt to the user's dynamic adjustments.

Although there are quite a few studies on root cause localization [8]–[11], unfortunately, these works are designed for specific applications and require training a separate model for each user, which is not suitable for cloud network scenarios. To the best of our knowledge, we conduct a comprehensive study of the root cause localization problem of shared bandwidth traffic anomalies in public cloud networks for the first time.

In this work, we propose *CloudPin*, a root cause localization framework for sBwp service traffic anomalies based on multidimensional analysis in public cloud networks. Specifically, we analyze the possible root cause VMs from two perspectives: absolute deviation and relative deviation. The absolute deviation is obtained by using the traffic time series prediction algorithm. The relative deviation is obtained from the analysis of the two dimensions, including anomaly amplitude and shape similarity. Furthermore, we comprehensively study the results from three dimensions of prediction deviation, anomaly amplitude, and shape similarity and propose an integrated ranking algorithm to obtain a ranking list of possible root cause VMs.

Our major contributions are summarized as follows:

- To the best of our knowledge, we conduct the first detailed analysis of root cause localization for sBwp service traffic anomalies in public cloud networks. To address this problem, we propose a root cause localization framework based on multi-dimensional analysis. Through a comprehensive analysis of prediction deviation, anomaly amplitude, and shape similarity, the framework can effectively address the challenge of traffic anomaly diversity in public cloud networks.
- We try different algorithms in the three dimensions of prediction deviation, anomaly amplitude, and shape similarity. By studying the results of different algorithms, we choose Moving Average, EVT-based, and set-based similarity algorithms, and propose a comprehensive ranking algorithm to integrate the results of the three dimensions. The algorithms we choose have the characteristics of low computational overhead and high real-time performance. Moreover, the algorithm framework can realize cold start to meet system dynamicity and storage load requirements.
- The system framework we propose is evaluated with real data in one of the world-renowned public cloud vendors. The evaluation results show that the performance of our proposed framework outperforms a large number of baseline algorithms.

The rest of this paper is organized as follows: In Section II, we introduce the background and motivations. In Section III, we describe the system design. In Section IV, we evaluate the real data in one of the world-renowned public cloud vendors. Related work is reviewed in Section V. We conclude the paper in Section VI.

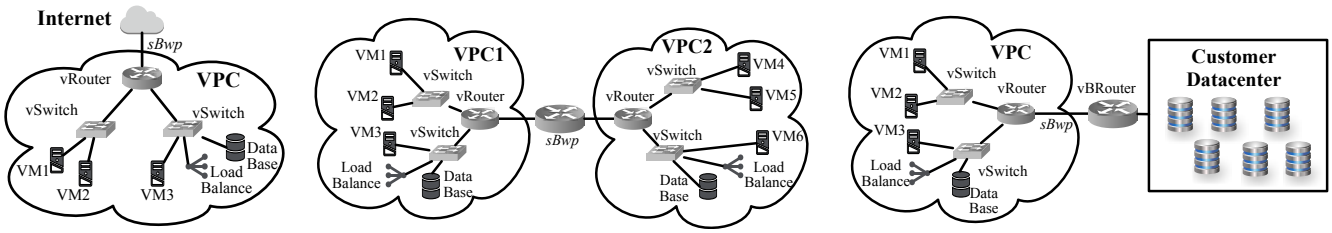
II. Background and Motivations

A. Application scenarios of sBwp in public cloud

There are plenty of application scenarios in the cloud network using the service of sBwp. We provide three frequently used application scenarios. Fig. 1(a) shows a typical virtual private cloud network (VPC) constructed by a user in the cloud network. A typical VPC usually includes several components, such as egress router (vRouter), virtual switches (vSwitches), virtual machines (VMs), load balancing (LB), and database (DB). Cloud users can deploy appropriate VPCs in the cloud network according to their business applications to provide external services over the Internet. To realize convenient management and low cost, users can choose to deploy sBwp service on vRouter so that all components in the VPC share sBwp resources. Some large enterprise users may need to deploy multiple VPCs in different regions, as shown in Figure 1(b). Cloud providers can provide high-speed communication services between different VPCs across regions. The sBwp service can be deployed on this high-speed channel, allowing components within VPCs in different regions to share communication bandwidth. Fig. 1(c) shows a hybrid cloud scenario, where users have both a local data center and a VPC on the cloud. To ensure the privacy of data, some enterprise users choose to put computing services on the cloud, but store the original data in the local data center. The cloud network can provide dedicated line services for hybrid cloud scenarios to realize the direct connection between the VPC on the cloud and the local data center through the virtual border gateway (vBRouter). For the dedicated line application, the cloud network can also provide sBwp services deployed on the vRouter in VPC to adopt all components in the VPC sharing sBwp resources.

B. Working process of the traffic anomaly localization

The working process of the traffic anomaly localization is shown in Fig. 2. (A) The user discovers an anomaly in the sBwp traffic time series on the user's egress gateway through visual observation or anomaly threshold warning. (B) The user reports the anomaly information to the cloud network operators in the form of a work order. The anomaly information generally includes region name, user ID, anomaly start time, duration time, and anomaly metric. Anomaly metrics generally include bytes per second (BPS) or packet per second (PPS). (C) After receiving the work order, the operator retrieves relevant data from the storage database according to the anomaly information. Then, the operator starts the anomaly root cause localization algorithm to analyze the obtained data. The output of the algorithm is a ranking list where the index is the VM ID, and the value is related to the root cause probability. The larger the value in the list, the more likely the root cause is. (D) The operator selects the results of top k



(a) Interaction with the Internet (b) Interaction between VPCs (c) Interaction with the customer Datacenter

Fig. 1. The application scenarios of the sBwp service.

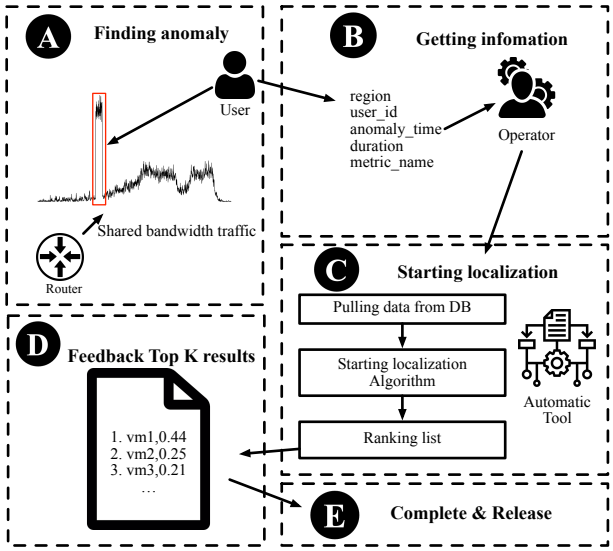


Fig. 2. Working process of the anomaly localization.

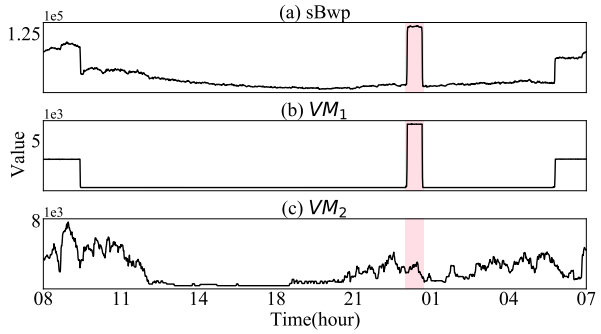


Fig. 3. An example of motivation.

and feeds them back to the user according to the result given by the algorithm. (E) The localization work is completed; the computing resources are released and the data cache is destroyed.

C. Motivations

Before introducing our motivation, we first show a specific anomaly example from the real cloud network environment, as shown in Fig. 3. Fig. 3(a) represents the traffic time series of the sBwp, and the shaded part is the anomaly time interval. This is a persistent spike anomaly lasting 40 minutes, which occurs frequently in cloud networks. Fig. 3(b) and Fig. 3(c) respectively denote the traffic time series of two VMs using this sBwp service. It can be seen that VM_1 has a similar anomaly when the sBwp is abnormal. However, VM_2 has no obvious change. Therefore, VM_1 is expected to be found by the localization algorithm, and VM_2 is expected to be excluded.

It can be seen from Fig. 3(a) and Fig. 3(b) that the anomaly is caused by a serious deviation from the expected normal value. Therefore, our intuitive idea is to build a traffic prediction model to determine the presence of anomalies by calculating the deviation between the prediction time series and the real time series, which is widely used in anomaly detection and root cause localization scenarios [12]. Specifically, we first calculate the deviation between the predicted and real value of the sBwp and VMs traffic time series during the anomaly time interval. Further, we divide the calculated deviation of each VM by the deviation of the sBwp to obtain the deviation ratio of each VM. Finally, we sort the deviation ratio of each VM. The higher the ranking, the more likely the root cause is.

However, this intuitive way has two limitations in practical deployment in large-scale and multi-scenario public cloud. First, current high-accuracy prediction algorithms are computation-intensive. Deep learning-based methods can achieve high prediction accuracy [13], but require a long offline training process to construct models. However, as the sBwp problem is user-specific, we cannot train all (millions of) users' data offline in advance, which results in an unaffordable computational cost. Second, we prefer cost-effective prediction algorithms in large-scale system, but they usually have low-accuracy. Traditional statistic-based methods, such as Difference [14], Moving Average [15], and Polynomial Regression [16], are simple and easy to implement, which have the advantages of fast training. However, since cloud services have diverse anomaly behaviors, a simple prediction algorithm is not likely to adapt to various scenarios.

To address the drawback of low accuracy in traditional statistical learning prediction methods, we add two new dimensions, anomaly amplitude and shape similarity. The prediction deviation is used to calculate the proportion of the absolute

deviation of each VM to the sBwp. When there are high-frequency fluctuations in the traffic time series, it is difficult for the prediction model to generate accurate results. In this scenario, we consider reducing the error of the prediction model by adding the analysis of the anomaly amplitude and the shape similarity. Specifically, the anomaly amplitude indicates the change amplitude of a VM traffic during the anomaly time interval compared to the normal time. When the change is more pronounced, it means that this VM is more likely to be the root cause. The shape similarity represents the similarity between the sequence shape of the VM's traffic time series and that of sBwp around the anomaly time interval. The higher the similarity, the more likely the VM is the root cause. Based on the calculation of these three dimensions, we further propose a ranking algorithm, which integrates the three dimensions to generate the final ranking result of root cause likelihood.

III. System Design

A. Overview

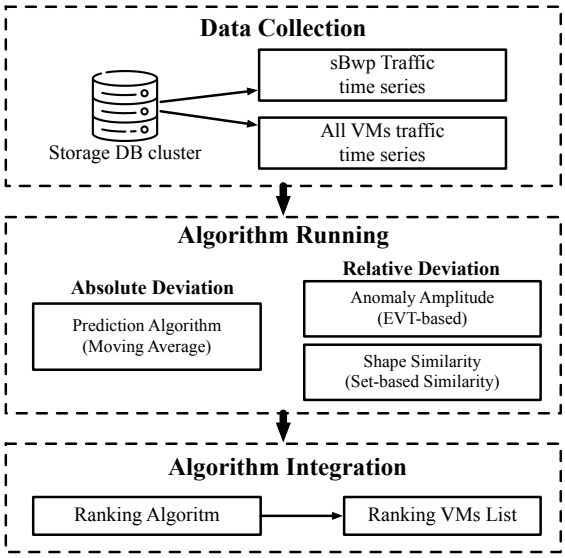


Fig. 4. Structure of localization system.

The structure of *CloudPin* is shown in Fig. 4. After the user reports the anomaly information, we first obtain the traffic data of the sBwp and all VMs using the sBwp service from the data storage cluster. To train the detection model, the data we obtain includes the data around the anomaly time interval and the data in the anomaly time interval. Specifically, we generally obtain the data 3 days before the anomaly start time and up to 8 hours after the anomaly end time (since some sensitive users may report the problem immediately when the anomaly occurs, the data after the anomaly end time may be less than 8 hours. In this case, we only use the data before the anomaly start time for training). After obtaining the traffic data, we respectively analyze the sBwp traffic time series and the VMs traffic time series in three dimensions. In the prediction deviation dimension, we design a model based on the Moving Average algorithm. In the dimension of anomaly amplitude, we use an improved model based on the EVT algorithm [17]. In the dimension of shape similarity, we select set-based similarity model [18]. After calculating the three dimensions, we design a ranking algorithm to integrate the results of the three dimensions and generate the final ranking list. Next, we explain the design of each model in detail.

B. Prediction model

Before describing the specific model, we first formalize the problem. Since there are many mathematical symbols in this paper, we summarize the meaning of the symbols in Table I. We define the traffic time series of a sBwp as $y(t)$, and the set of traffic time series of the VMs is $X = \{x_1(t), x_2(t), ..., x_n(t)\}$, where $x(t)$ represents the traffic time series of a VM, n represents the number of VMs. Since all $x(t)$ in X together produce the sBwp traffic $y(t)$, $y(t) = x_1(t)+x_2(t)+...+x_n(t)$. We define the anomaly time interval as (t_s, t_e), where t_s is the start time of the anomaly and t_e is the end time. The goal of the localization algorithm is to find a subset of X that is the anomaly root cause of $y(t)$ in (t_s, t_e).

TABLE I
The meaning of the mathematical symbols.

Meaning	Symbol	Meaning	Symbol	Meaning	Symbol
sBwp time series	$y(t)$	VM time series set	X	VM time series	$x(t)$
Anomaly start time	t_s	Anomaly end time	t_e	Prediction Model	H
Absolute deviation at one time point	d_f	Total absolute deviation	d_f^{sum}	Ratio of the cumulative absolute deviation	d_f^p
Threshold of EVT algorithm	θ	Anomaly amplitude at one time point	α	Anomaly amplitude of time series	α^{max}
Shape similarity of the $y(t)$ and $x(t)$	$S(y,x)$	Discretization set of time series	Z	Final result of integrated algorithm	In

To quantify the anomalies generated by $y(t)$ and X in the (t_s, t_e), we first use a prediction model to calculate the absolute deviation. Specifically, for the $y(t)$, we use a prediction model H, and $H(y(t_c))$ represents the predicted value of $y(t)$ at time t_c. We define the absolute deviation of $y(t)$ at t_c as $d_f(y(t_c)) = |y(t_c) - H(y(t_c))|$. For the absolute deviation of the (t_s, t_e) interval, we use the cumulative sum method to sum the d_f in the (t_s, t_s) to obtain the total absolute deviation of $y(t)$ in the (t_s, t_e) as $d_f^{sum}(y(t)) = d_f(y(t_s))+d_f(y(t_s+1))+...+d_f(y(t_e))$. Similarly, we calculate the cumulative absolute deviation in (t_s, t_e) for each $x(t)$ in the set X as $d_f^{sum}(x_1(t)), d_f^{sum}(x_2(t)), \ldots, d_f^{sum}(x_n(t))$. Furthermore, we obtain the ratio of the cumulative absolute deviation of each VM to the sBwp as $d_f^p(x(t)) = \frac{d_f^{sum}(x(t))}{d_f^{sum}(y(t))}$, where $d_f^p(x(t)) \in [0, 1]$ (Since $y(t)$ is the sum of all $x(t)$, generally $d_f^p(x(t)) \leq 1$. However, it is also possible that due to the uncertainty of the prediction model, $d_f^p(x(t))$ may be slightly greater than 1. Our algorithm does not strictly require

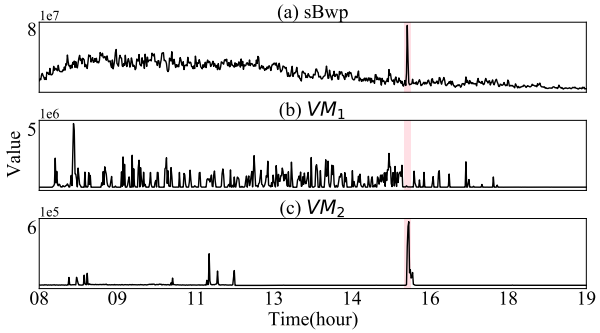


Fig. 5. A false positive example of prediction model.

that $d_f^p(x(t))$ must be less than 1, hence this special case will not affect the accuracy of the algorithm). Finally, we sort the absolute deviation percentage of each VM $d_f^p(x(t))$ in descending order. The higher the value of $d_f^p(x(t))$, the more likely it is the root cause.

Next, we introduce the choice of prediction model H. As described in Section II, prediction models based on deep learning are not suitable for large-scale and complex scenarios in public cloud networks. Therefore, we choose to use statistical learning methods. We compare several algorithms and select the Moving Average algorithm with the best performance in all prediction models, the detail results shown in Section IV-B. We analyze this because when the anomaly occurs, the abnormal data points have significant differences compared with the surrounding normal data points. Therefore, the Moving Average algorithm can fit the surrounding normal data well to show the difference between the abnormal points and the normal points. Specifically, we maintain a sliding window of length m and calculate the average value of all points in the sliding window as the predicted value of t_s. Next, we move the sliding window M to obtain the predicted value at each moment in (t_s, t_e) and the accumulated deviation. In practice, we compare different sliding window lengths and find that $m = 60$ minutes is an optimal parameter.

C. Anomaly amplitude

In most cases, the prediction model can quantify the absolute deviation of the traffic time series very well. However, in the scenario of high-frequency jitter traffic time series, it is difficult for the prediction model to generate effective absolute deviations directly. For example, Fig. 5 shows a false positive example of the prediction model. As can be seen from Fig. 5(a), the shaded part indicates an obvious transient spike in the sBwp traffic time series. Fig. 5(b) shows the traffic time series of the VM with the largest absolute deviation given by the prediction model. However, it is a false positive, and the absolute deviation ratio is 0.025. Fig. 5(c) shows one of the actual root cause VMs. The absolute deviation ratio of Fig. 5(c) calculated by the prediction model is 0.014. Therefore, it can be seen that for the high-frequency jitter traffic time series, similar to that in Fig. 5(b), it is challenging to generate correct results using only the prediction model.

To address this problem, we add the dimension of the anomaly amplitude in the relative deviation. The absolute deviation generated by the prediction model mainly considers the deviation ratio of each VM to the sBwp traffic time series within the anomaly time interval. Unlike the absolute deviation, the anomaly amplitude considers the anomaly significance of each VM's traffic time series at the anomaly time compared with its own normal time. For example, the anomaly amplitude of the traffic time series in Fig. 5(b) is relatively low because it does not significantly change at the anomaly time interval compared with the surroundings. On the contrary, the anomaly amplitude in Fig. 5(c) is higher since there is a abrupt increase at the anomaly time interval compared with the surroundings.

To quantify anomaly amplitude, we compare two different algorithms. One is the non-parametric estimation method based on KDE [19], and the other is the EVT algorithm based on extreme value theory [17]. We choose the EVT algorithm based on extreme value theory since it has better performance (detailed experimental results are discussed in Section IV-B). Next, we briefly introduce the EVT algorithm, and a detailed description refers to [17]. EVT is an effective peak detection algorithm since it infers the extreme value distribution in the data, rather than making strong assumptions on the original data.

Given a random variable W, and its cumulative probability density function $F(w) = P(W \leq w)$. We define $\overline{F}(w) = 1 - F(w) = P(w > W)$, which means the 'tail' of the data distribution. According to existing work, Tippett [20] and later Gnedenko [21], $\overline{F}(w)$ generally has the following form:

$$G_\gamma \ : \ w \mapsto exp(-(1+\gamma w)^{-\frac{1}{\gamma}})), \ \gamma \in R, \ 1+\gamma w > 0.$$

γ is the extreme value index that depends on the original law. Because we lack assumptions about the initial distribution, γ is difficult to choose directly. To effectively solve this problem, Pickands-Balkema-De Haan theorem [22], [23] is proposed as follows:

Theorem 1 (Pickands-Balkema-De Haan): The cumulative distribution function $F \in \mathcal{D}_\gamma$ if and only if a function σ exists, for all $w \in R \ s.t. \ 1+\gamma w > 0$:

$$\overline{F}_t(w) = P(W - t > w \mid W > t) \underset{t \to \tau}{\backsim} (1 + \frac{\gamma w}{\sigma(t)})^{-\frac{1}{\gamma}}$$

This result shows that the excess over a threshold t, written $W - t$, are likely to follow a Generalized Pareto Distribution (GPD) with parameters γ, σ. And t is a initial 'high' threshold. Based on domain experience and [17], we set t to a high empirical percentile (say 98%). And then like the method in [17], we use the method of Maximum Likelihood Estimation to estimate the σ and γ. The output of the EVT algorithm is the anomaly threshold. Anomalies in the cloud network include spike and trough, therefore we use a double-direction EVT algorithm, that means EVT algorithm outputting a maximum anomaly threshold θ_h, and a minimum threshold θ_l. Further, we define the anomaly amplitudes of the traffic time series $x(t)$ at time t_c as $\alpha_h(t_c)$ and $\alpha_l(t_c)$:

$$\alpha_h(t_c) = \begin{cases} \dfrac{x(t_c) - \theta_h}{x(t_c)}, & if\ x(t_c) > \theta_h, \\ 0, & if\ x(t_c) \leq \theta_h. \end{cases}$$

$$\alpha_l(t_c) = \begin{cases} \dfrac{\theta_l - x(t_c)}{\theta_l}, & if\ x(t_c) < \theta_l, \\ 0, & if\ x(t_c) \geq \theta_l. \end{cases}$$

$\alpha_h(t_c)$ and $\alpha_l(t_c)$ respectively represent the spike and trough anomaly amplitude of the traffic time series $x(t)$ at time t_c. Since the anomaly time lasts from t_s to t_e, we define the anomaly amplitudes in the time range (t_s, t_e) as:

$$\alpha_h^{max} = max(\alpha_h(t)),\ for\ all\ t\ in\ (t_s, t_e)$$

$$\alpha_l^{max} = max(\alpha_l(t)),\ for\ all\ t\ in\ (t_s, t_e)$$

α_h^{max} and α_l^{max} respectively represent the spike and the trough anomaly amplitude of the traffic time series in the time interval (t_s, t_e). And, $\alpha_h^{max}, \alpha_l^{max} \in [0, 1]$, the closer to 1, the more apparent anomaly. For example, α_h^{max} of the traffic time series in Fig. 5(b) is 0, but α_h^{max} in Fig. 5(c) is 0.85. Therefore, it can be seen that the anomaly amplitude can be used to effectively detect the anomaly in Fig. 5(c). We introduce in detail how the anomaly amplitude is integrated with the prediction model in Section III-C.

D. Shape similarity

Besides the anomaly amplitude, from the perspective of relative deviation, we also need to consider the shape similarity between VM and sBwp traffic sequence in the anomaly time interval to solve the challenge of cloud network traffic complexity. For example, the traffic time series in Fig. 3(a) and Fig. 3(b) have similar shapes in the anomaly time interval.

At present, there are many research works related to sequence shape similarity. The method based on distance is prevalent, such as DTW algorithm [24]. Although the DTW calculation method has a high accuracy, its high computational complexity is not suitable for a large-scale cloud network environment. Since we only need to analyze root cause and do not have very high accuracy requirements for similarity, we choose a set-based similarity method, which has a good trade-off between accuracy and efficiency, and is one of the state-of-art method [18]. We briefly describe the set-based similarity method, and details refer to [18]. The core idea based on the set similarity algorithm is to convert the continuous time series similarity problem to the discrete set similarity problem. The specific calculation process is as follows:

Given the sBwp traffic time series $y(t)$, the traffic time series of a VM $x(t)$, and the anomaly time interval (t_s, t_e). The goal of similarity algorithm is to calculate the shape similarity $S(y, x)$ between $y(t)$ and $x(t)$ around the interval (t_s, t_e).

First, since $y(t)$ and $x(t)$ may have difference in magnitude of the values, we normalize $y(t)$ and $x(t)$ by commonly used approach z-normalization [25]. To reflect the difference between the data during the anomaly time and the surrounding normal time, we extend the normalization time interval with a Δt time interval around the anomaly time interval, $(t_s - \Delta t, t_e + \Delta t)$. According to operating experience, we choose Δt as 1 hour. The specific calculations are as follows:

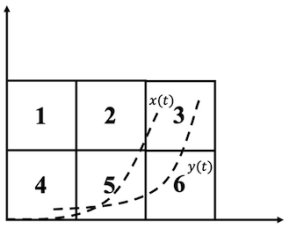


Fig. 6. A example of grid division.

$$Norm(y(t)) = \frac{y(t) - mean(y(t))}{std(y(t))}$$

$$Norm(x(t)) = \frac{x(t) - mean(x(t))}{std(x(t))}$$

Mean and std respectively represent the mean and standard deviation of the traffic time series in $(t_s - \Delta t, t_e + \Delta t)$. Next, we introduce the process of time series discretization. First, we cut the coordinate plane into $k * l$ cells, as shown in Fig. 6. k represents the number of grids on the horizontal axis, and l represents the number of grids on the vertical axis. For example, $k = 3$, $l = 2$ in Fig. 6. Further, we sequentially number the cells as shown in Fig. 6. We define the set of cell numbers that each sequence passes through as the discretization representation Z of the sequence. For example, the discretization set of $y(t)$ in Fig. 6 is represented as $Z(y(t)) = \{3, 4, 5, 6\}$ and the discretization set of $x(t)$ is represented as $Z(x(t)) = \{2, 3, 4, 5\}$. After discretization representation, we use jaccard metric to calculate the similarity of discretization sets [26], shown as follows:

$$Jaccard(Z(y(t)), Z(x(t))) = \frac{|Z(y(t)) \cap Z(x(t))|}{|Z(y(t)) \cup Z(x(t))|}$$

This method can effectively calculate the shape similarity between the VM and sBwp traffic time series. For example, the shape similarity of the time series in Fig. 3(a) and Fig. 3(b) is 0.90. However, the shape similarity of Fig. 3(a) and Fig. 3(c) is only 0.15.

E. Ranking algorithm

When the calculation of the above three dimensions is completed, we need to integrate the results of the three dimensions. The commonly used integration algorithm is a simple weighted average [27]. But this method is not suitable for our problem since the deviation perspectives described by the three dimensions are different. Therefore, we need to design a reasonable integration algorithm. The prediction model describes the absolute deviation, which is the basis of root cause analysis. The anomaly amplitude and shape similarity characterize the relative deviation, which is a further

description on the basis of the absolute deviation. Therefore, the integration algorithm we designed is as follows:

$$In(x(t)) = d_f^p(x(t)) * (\omega_\alpha * \alpha^{max}(x(t)) + \omega_s * S(x(t)))$$

$In(x(t))$ represents the final result of VM traffic time series $x(t)$ after the integration. d_f^p, α^{max}, S respectively represent the result of the prediction model, anomaly amplitude, and shape similarity of $x(t)$. ω_α and ω_s respectively represent the weights of anomaly amplitude and shape similarity. Through this integration algorithm, it ensures that on the basis of absolute deviation, the analysis of relative deviation can be effectively reflected.

Next, we discuss the calculation method of the anomaly amplitude α^{max}. In Section III-E, we calculate the anomaly amplitudes of spike and trough separately. Therefore, in the integration algorithm, there are different considerations for the calculation of α^{max}. In practice, we first calculate the anomaly amplitudes of the spike and trough of the sBwp traffic time series $y(t)$. If $\alpha_h^{max}(y(t)) > 0, \alpha_l^{max}(y(t)) = 0$, which means that this is a spike anomaly. Therefore, we only calculate α_h^{max} of VM as anomaly amplitude. On the contrary, if the anomaly amplitude of the sBwp is $\alpha_h^{max}(y(t)) = 0, \alpha_l^{max}(y(t)) > 0$; we only calculate the α_l^{max} of the VM. When the anomaly amplitude of sBwp is $\alpha_h^{max}(y(t)) > 0, \alpha_l^{max}(y(t)) > 0$, this indicates that the sBwp may have a jitter anomaly. Therefore, we calculate the α_h^{max} and α_l^{max} of the VM separately, and then $\alpha^{max} = (\alpha_h^{max} + \alpha_l^{max})/2$. The last special case is $\alpha_h^{max}(y(t)) = 0, \alpha_l^{max}(y(t)) = 0$ for sBwp. This case indicates that the sBwp does not produce a significant increase or decrease. Therefore, the anomaly amplitude cannot generate effective results in this case. We set $\alpha^{max} = 0$ for all VMs, which means that the effect of anomaly amplitude is ignored.

Finally, we discuss the choice of weights ω_α and ω_s. ω_α and ω_s respectively represent the weight of anomaly amplitude and shape similarity in root cause analysis. The weight setting strategy is different for different users due to different business applications. Therefore, users can choose appropriate weights according to their applications. We set $\omega_\alpha = \omega_s = 0.5$ by default, which means anomaly amplitude and shape similarity have the same importance. We further discuss the impact of weight settings on algorithm performance in Section IV-E.

IV. Evaluation

A. Setup

In our evaluation, we select 3 data centers from a world-renowned cloud network vendor. Through the observation of the operator and the work orders reported by users, we find a total of 183 anomaly cases in these three data centers, from 2020-11-30 to 2021-04-01, about four months. The details of the data are shown in Table II. Taking data center A as an example, we explain the meaning of the data in the table. We find 77 anomaly sBwp cases in data center A. The total number of machines using these bandwidth packages is 24,549. Since the number of machines in each sBwp is different, we count the distribution of the number of machines in sBwp. Among the 77 anomaly cases in data center A, 76 sBwp services have more than ten machines; 8 sBwp services have more than 1,000 machines, and the largest one has 2,483 machines. We collect traffic time series of these sBwp and VMs in three data centers from the centralized data storage center on an analysis server. In the annotation of datasets, it is not feasible to mark all VMs manually. Therefore, we adopt a method similar to [9]. We consider the union of root causes detected by all baseline algorithms as candidate root cause sets, and then verify them manually by experienced experts one by one. Using this method to evaluate the algorithm can obtain accurate precision, but the recall may be biased (because the root cause may be missed by all the above methods), hence we only choose precision as the evaluation metric when evaluating the algorithm performance.

Since the output of our algorithm is a ranking list, we choose precision@top k as the evaluation metric. According to the processing experience of the operation, we choose the maximum value of k as 5, which means that we prioritize the investigation of the first five possible root causes generated by the algorithm.

TABLE II
Details of the data.

Region	#cases	#machines	virtual machines distribution					
			> 10	> 50	> 100	> 500	> 1000	> 10000
A	77	24549	76	54	25	8	8	0
B	49	50815	47	41	23	11	10	1
C	57	9287	57	53	25	2	1	0

B. Overall evaluation

First, we evaluate the overall performance of the algorithm. Our algorithm uses three-dimensional analysis and an integrated model. Therefore, our baseline algorithm includes four aspects: the simple model of prediction deviation, anomaly amplitude, shape similarity, and the multi-dimensional integrated model. We select the parameters of baseline models that perform best in our cases through plenty of attempts.

Baseline prediction models include Linear Regression [28], Olympic (native seasonal model), Simple Exponential Smoothing [16] , Polynomial, Difference [14], Moving Median and Moving Average.

We choose two anomaly amplitude baseline models, as described in Section III-C, one is the EVT-based algorithm we use, and the other is the non-parametric estimation algorithm based on KDE [19]. The KDE algorithm infers the probability density distribution function of the original sequence through non-parametric estimation of the traffic sequence. Then we obtain the anomaly probability as the anomaly magnitude through the p-value method.

The shape similarity baseline models include the set-based similarity algorithm we use and the spearman correlation coefficient method [29].

The integrated models include simple Top algorithm and EGADS algorithm [30]. The Top algorithm is a traditional method used by operators, representing a simple ranking based

TABLE III
OVERALL PERFORMANCE EVALUATIONS.

Type	Algorithm name	Precision@1	Precision@2	Precision@3	Precision@4	Precision@5	Average precision
Prediction Model	Linear Regression	0.587	0.570	0.579	0.579	0.589	0.581
	Olympic	0.847	0.847	0.839	0.837	0.833	0.841
	Polynomial	0.579	0.552	0.560	0.563	0.559	0.563
	Difference	0.885	0.869	0.852	0.839	0.834	0.856
	Moving Median	0.568	0.568	0.564	0.565	0.570	0.567
	Moving Average	0.891	0.869	0.856	0.846	0.844	0.861
	Simple Exponential Smoothing	0.770	0.779	0.779	0.776	0.774	0.776
Anomaly Amplitude	KDE	0.508	0.481	0.484	0.483	0.473	0.486
	EVT	0.656	0.653	0.650	0.651	0.648	0.652
Shape Similarity	Set-based	0.689	0.667	0.656	0.646	0.637	0.659
	Spearman	0.650	0.626	0.612	0.599	0.598	0.617
Integrated Model	Top	0.776	0.743	0.741	0.749	0.746	0.751
	EGADS	0.825	0.795	0.784	0.780	0.779	0.793
	CloudPin	**0.978**	**0.918**	**0.902**	**0.893**	**0.884**	**0.915**

on traffic statistics. The EGADS algorithm is an anomaly detection algorithm proposed by yahoo [30]. EGADS algorithm tries to solve the problem that a single prediction model is challenging to provide accurate predictions. It uses the results of multiple prediction models to compare in the training data set, and select the model with the lowest prediction deviation as the detection model.

The performance evaluation results of the above algorithm are shown in Table III. First, it can be seen from the overall algorithm performance comparison that *CloudPin* achieves the best performance under different k. Especially when k=1, which is the most likely root cause, *CloudPin* can reach precision of 97.8%, far exceeding other baseline algorithms. In actual operation, the root cause of ranking number one is often the most concerned by users, and this is the highest priority for users to analyze application issues. Then, compared with the anomaly amplitude and shape similarity algorithm, the prediction model algorithms have relatively better performance. For example, the average precision of Olympic, Difference, and Moving average can reach more than 80%. The average precision of the anomaly amplitude and shape similarity algorithm are below 70%. This shows that the absolute deviation produced by the prediction model is the basis of root cause analysis, while the relative deviation mainly plays a role in auxiliary analysis. This also verifies the rationality of our ranking algorithm model. In addition, the moving average algorithm, the EVT algorithm and the Set-based similarity algorithm achieve the best performance in their respective dimensions, which validates the basis for our model selection.

For the integrated model, it can be seen that the traditional Top algorithm is obviously inferior. This is because the Top algorithm cannot solve some cases with high changes or declines. Although the EGADS algorithm integrates a variety of prediction models, EGADS is not as good as Moving Average. This shows that it is difficult to locating root cause only through the optimization of prediction models. We speculate that this may be due to the fact that EGADS adopts different algorithms and selects the best model on the training data set. It is difficult to ensure that the selected prediction model is still optimal in the anomaly time interval. Therefore, the overall performance of EGADS is not significantly improved compared to the single model.

C. Data center evaluation

To verify the robustness of the algorithms in different data centers, we respectively evaluate the data in the three data centers from different regions, and the results are shown in Fig. 7. We can see that our algorithm outperforms other baseline algorithms in different data centers. Although all algorithms in data center C have better performance due to the obvious anomalies in data center C. The overall performance of different algorithms in different data centers is close. Therefore, the location of data centers has no significant impact on the performance of the algorithm. We analyze that this is mainly because users and applications in different data centers are vast. There are many overlaps between different data centers.

D. Anomaly type evaluation

As mentioned in the introduction, the diversity of anomaly types is a core challenge in cloud networks. By observing the sBwp data in cloud networks, the common types of anomaly cases can be divided into five parts, including persistent spike, transient spike, persistent trough, transient trough, and others, some of examples as shown in Fig. 8. According to operation experience, we commonly define an anomaly exceeding 30 minutes as a persistent anomaly. Conversely, an anomaly, duration time less than 30 minutes, is defined as the transient anomaly. Others refer to anomalies that are not within the scope of spike or trough and commonly refer to some high-frequency jitter anomalies. Through the above definition, we manually label the anomaly types of all cases.

To verify the generality of the algorithms for different anomaly types, we evaluate the different types of anomaly cases, as shown in Fig. 9. First, it can be seen that in different types of cases, the performance of *CloudPin* exceeds other baseline algorithms. For anomalies other than Others,

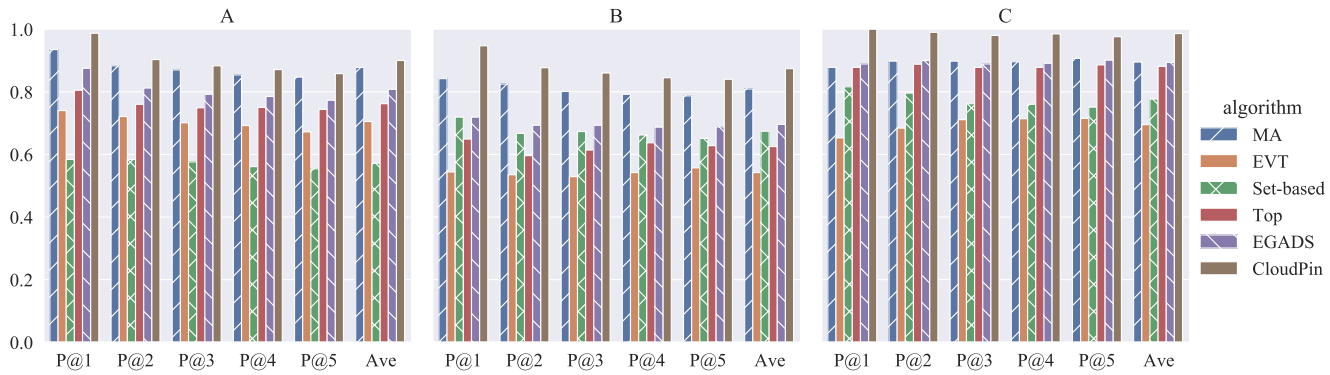


Fig. 7. Performance of different regions.

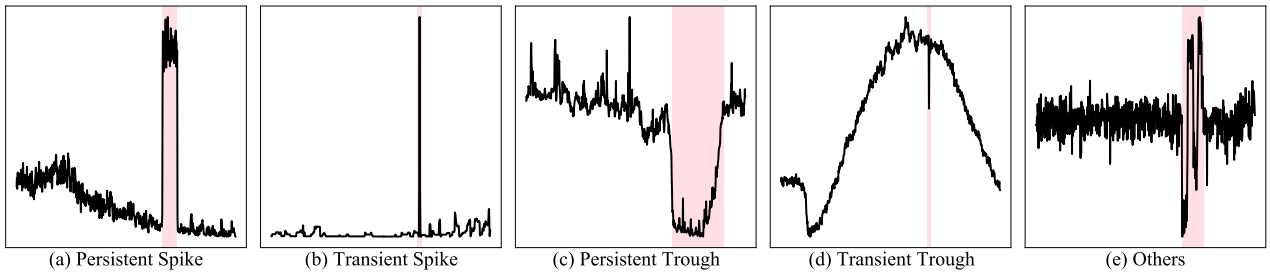


Fig. 8. Examples of different anomaly classification.

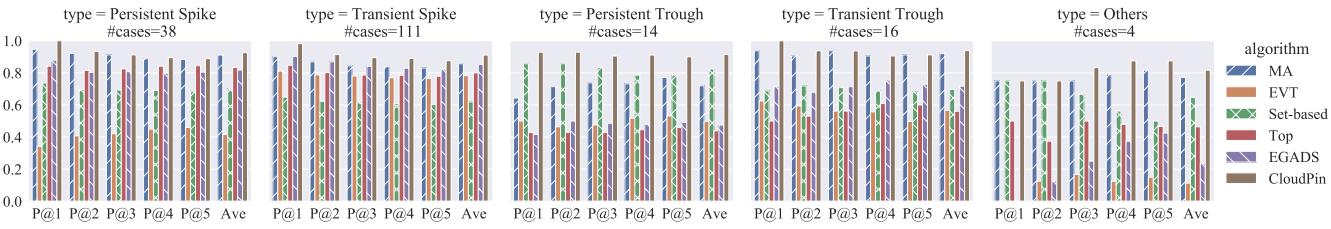


Fig. 9. Performance evaluations of different anomaly types.

we can see that the traditional Top algorithm has relatively good performance in spike scenarios and poor performance in trough scenarios. This is consistent with what we described in the Section IV-B that the Top algorithm could not solve the trough scenario. The Moving Average algorithm based on prediction deviation has good performance in transient spike and trough but not effective in persistent anomalies. This is mainly because the prediction model has a cumulative effect on long-term prediction errors. The EVT-based anomaly amplitude algorithm has better results in transient anomalies than persistent anomalies, which is similar to the prediction model. This is mainly because as the time interval becomes longer, the number of interferential points may increase, leading to the EVT algorithm generating errors. The set-based shape similarity model performs better in persistent anomalies compared with transient anomalies. This is because the longer the anomaly duration time, the more obvious the sequence shape becomes. Due to the limited sample number of Others, it is difficult to give a statistically significant result. From the existing 4 cases, it can be seen that the average precision of *CloudPin* still has a slight improvement compared with other algorithms.

TABLE IV
PERFORMANCE OF DIFFERENT WEIGHTS.

ω_α	ω_s	P@1	P@2	P@3	P@4	P@5	Ave
0.0	1.0	0.951	0.902	0.880	0.871	0.861	0.893
1.0	0.0	0.962	0.910	0.898	0.890	0.878	0.907
0.5	0.5	0.978	0.918	0.902	0.893	0.884	0.915

TABLE V
CONSUMPTION TIME EVALUATION.

#VM	#cores	Algorithm name	Time	Speed
1994	1	Moving Average	223s	8.9 vm/s
		EVT	294s	6.8 vm/s
		Set-based	175s	11.4 vm/s
		EGADS	11263s	0.2 vm/s
		CloudPin	**695s**	**2.9 vm/s**
	16	**CloudPin**	**49s**	**40.7 vm/s**

E. Ranking weight discussion

The integration algorithm we designed needs to set the weight of anomaly amplitude and shape similarity. Although the above evaluation experiments show that our default settings ($\omega_\alpha = 0.5, \omega_s = 0.5$) can achieve effective results. We further discuss the influence of the weight setting on the performance of the algorithm, and the evaluation results are shown in Table IV. ($\omega_\alpha = 0.0, \omega_s = 1.0$) means that the algorithm only calculates the relative deviation caused by the shape similarity, and ($\omega_\alpha = 1.0, \omega_s = 0.0$) means that the algorithm only considers the relative deviation caused by the anomaly amplitude. We can see that when only the relative deviation of shape similarity or anomaly amplitude is considered, the performance of the algorithm is not as good as the default configuration. This shows that the anomaly amplitude and shape similarity are both meaningful in the algorithm. For specific cases, users can adjust the weight settings appropriately according to their application and anomaly types. For example, for persistent anomalies, the weight of the shape similarity ω_s can be appropriately increased, while for transient anomalies, the weight of the anomaly amplitude ω_α can be appropriately increased.

F. Overhead evaluation

The overhead evaluation of the algorithm includes storage space and calculation time. Because the algorithm only uses simple metrics, such as PPS, BPS, etc., and the training data only needs about three days, the average storage space of each traffic time series does not exceed 200KB. Taking the largest case of our collected data as an example, the storage overhead used by the user with 33,601 VMs is about $200KB * 33,601 \approx 7GB$. This storage overhead is several orders of magnitude lower than that of flow analysis methods such as Netflow [31]. Therefore, the storage overhead of our algorithm is almost negligible.

We evaluate the calculation time of a case with 1994 VMs on an ordinary server (CPU: 16 cores, Memory: 128GB DDR4). The evaluation results are shown in Table V. We can see that in the case of a single core, the calculation time of the three dimensions alone is between 150-300s, and there is no obvious difference in orders of magnitude. This shows that our three-dimensional calculation method is appropriate in terms of computing time avoiding the situation of one sub-model waiting another for a long time. In addition, our integrated algorithm is significantly faster than the EGADS algorithm. This is mainly because EGADS uses a large number of prediction models, and we only use a single prediction model. We solve the problem of inaccuracy of a single prediction model by adding two relative deviation dimensions to make the number of models controllable. In addition, because *CloudPin* can perform parallel calculations, consumption time can be reduced to 49 seconds when all 16 CPU cores are used. According to our practical experience, manual fault location often takes several hours due to excessive disposal processes. Therefore, the consumption time of *CloudPin* is acceptable in the operation and maintenance of cloud networks.

V. Related Work

Specific-application root cause analysis. Although there are many previous works on root cause localization. Most of these works are designed for a specific application software system. Some works [8], [32]–[36] rely on the dependency topology of the service. For example, MicroRCA [8] needs to construct an attributed graph with services and hosts to model the anomaly propagation among services. However, the cloud provider cannot directly obtain the user's application topology information in a cloud network. Except for using topology, TraceAnomaly uses microservice invocation traces to detect anomalies [9]. MEPFL tries to locate microservice failures by learning system trace logs [37]. However, cloud network providers are unable to obtain these specific application operating data subject to user privacy. In addition, Fluxrank constructs a framework to localize the root cause machines by using machine-level indicators, including the network metric used by us [10]. However, this method requires building a separate model for each user in the cloud network. Hence this high overhead is unacceptable for the cloud network operator.

Time series anomaly detection algorithm. Commonly, time series anomaly detection algorithms include density-based, clustering-based, prediction-based, and so on [12]. Since the prediction-based model can quantify the anomaly, we adopt the prediction-based method. Prediction models include traditional statistical learning methods and deep learning methods. The statistical methods have been discussed in detail as baseline algorithms. We provide the works of prediction model based on deep learning. LSTM is the most commonly used neural network model for time series prediction and anomaly detection [13], [38]–[40]. Although the LSTM-based methods have high accuracy, they need to train an offline model separately for each user, which is high overhead and inflexible against the requirements of the cloud network.

VI. Conclusion

In this paper, we propose *CloudPin*, a framework based on multi-dimensional analysis to locate root cause of sBwp traffic anomalies in public cloud networks. Our algorithm analyzes the three dimensions of prediction deviation, anomaly amplitude, and shape similarity and generates possible root lists through an integrated ranking algorithm. We conduct a comprehensive evaluation in a real large-scale cloud network.

VII. Acknowledgment

Shize Zhang, Yunfeng Zhao, Zhiliang Wang, Jiahai Yang, Lin He, Jianping Wu are with the Institute for Network Sciences and Cyberspace, Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084. This work is supported by the National Key Research and Development Program of China under Grant No.2018YFB1800204, Alibaba Group through Alibaba Innovative Research Program, the Key-Area Research and Development Program of Guangdong Province (2019B010136001), the Science and Technology Planning Project of Guangdong Province LZC0023.

REFERENCES

[1] "Amazon web services (aws)," https://aws.amazon.com/.

[2] "Microsoft azure," https://azure.microsoft.com/.

[3] "Alibaba cloud," https://www.alibabacloud.com/.

[4] "Aws vs azure vs google cloud market share 2021: What the latest data shows," https://www.parkmycloud.com/blog/aws-vs-azure-vs-google-cloud-market-share/, 2021.

[5] G. J. Popek and R. P. Goldberg, "Formal requirements for virtualizable third generation architectures," *Communications of the ACM*, vol. 17, no. 7, pp. 412–421, 1974.

[6] K. Qian, S. Ma, M. Miao, J. Lu, T. Zhang, P. Wang, C. Sun, and F. Ren, "Flexgate: High-performance heterogeneous gateway in data centers," in *Proceedings of the 3rd Asia-Pacific Workshop on Networking 2019*, 2019, pp. 36–42.

[7] "5 best technologies to build microservices architecture," https://www.clariontech.com/blog/5-best-technologies-to-build-microservices-architecture, 2021.

[8] L. Wu, J. Tordsson, E. Elmroth, and O. Kao, "Microrca: Root cause localization of performance issues in microservices," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–9.

[9] P. Liu, H. Xu, Q. Ouyang, R. Jiao, Z. Chen, S. Zhang, J. Yang, L. Mo, J. Zeng, W. Xue *et al.*, "Unsupervised detection of microservice trace anomalies through service-level deep bayesian networks," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2020, pp. 48–58.

[10] P. Liu, Y. Chen, X. Nie, J. Zhu, S. Zhang, K. Sui, M. Zhang, and D. Pei, "Fluxrank: A widely-deployable framework to automatically localizing root cause machines for software service failure mitigation," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2019, pp. 35–46.

[11] Y. Su, Y. Zhao, W. Xia, R. Liu, J. Bu, J. Zhu, Y. Cao, H. Li, C. Niu, Y. Zhang *et al.*, "Coflux: robustly correlating kpis by fluctuations for service troubleshooting," in *Proceedings of the International Symposium on Quality of Service*, 2019, pp. 1–10.

[12] H.-S. Wu, "A survey of research on anomaly detection for time series," in *2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. IEEE, 2016, pp. 426–431.

[13] T. Ergen and S. S. Kozat, "Unsupervised anomaly detection with lstm neural networks," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 8, pp. 3127–3141, 2019.

[14] R. Chrobok, O. Kaumann, J. Wahle, and M. Schreckenberg, "Different methods of traffic forecast based on real data," *European Journal of Operational Research*, vol. 155, no. 3, pp. 558–568, 2004.

[15] S. Hansun, "A new approach of moving average method in time series analysis," in *2013 conference on new media studies (CoNMedia)*. IEEE, 2013, pp. 1–4.

[16] E. Ostertagová and O. Ostertag, "The simple exponential smoothing model," in *The 4th International Conference on Modelling of Mechanical and Mechatronic Systems, Technical University of Košice, Slovak Republic, Proceedings of conference*, 2011, pp. 380–384.

[17] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet, "Anomaly detection in streams with extreme value theory," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1067–1075.

[18] J. Peng, H. Wang, J. Li, and H. Gao, "Set-based similarity search for time series," in *Proceedings of the 2016 International Conference on Management of Data*, 2016, pp. 2039–2052.

[19] R. Laxhammar, G. Falkman, and E. Sviestins, "Anomaly detection in sea traffic-a comparison of the gaussian mixture model and the kernel density estimator," in *2009 12th International Conference on Information Fusion*. IEEE, 2009, pp. 756–763.

[20] R. A. Fisher and L. H. C. Tippett, "Limiting forms of the frequency distribution of the largest or smallest member of a sample," in *Mathematical proceedings of the Cambridge philosophical society*, vol. 24, no. 2. Cambridge University Press, 1928, pp. 180–190.

[21] B. Gnedenko, "Sur la distribution limite du terme maximum d'une serie aleatoire," *Annals of mathematics*, pp. 423–453, 1943.

[22] A. A. Balkema and L. De Haan, "Residual life time at great age," *The Annals of probability*, pp. 792–804, 1974.

[23] J. Pickands III *et al.*, "Statistical inference using extreme order statistics," *Annals of statistics*, vol. 3, no. 1, pp. 119–131, 1975.

[24] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, USA:, 1994, pp. 359–370.

[25] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[26] M. Levandowsky and D. Winter, "Distance between sets," *Nature*, vol. 234, no. 5323, pp. 34–35, 1971.

[27] L. A. Alexandre, A. C. Campilho, and M. Kamel, "Combining independent and unbiased classifiers using weighted average," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 2. IEEE, 2000, pp. 495–498.

[28] P. Högberg, "Estimation of parameters in models for traffic prediction: a non-linear regression approach," *Transportation Research*, vol. 10, no. 4, pp. 263–265, 1976.

[29] L. Myers and M. J. Sirois, "S pearman correlation coefficients, differences between," *Encyclopedia of statistical sciences*, 2004.

[30] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1939–1947.

[31] B.-Y. Choi and S. Bhattacharyya, "Observations on cisco sampled netflow," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 3, pp. 18–23, 2005.

[32] J. Weng, J. H. Wang, J. Yang, and Y. Yang, "Root cause analysis of anomalies of multitier services in public clouds," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1646–1659, 2018.

[33] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang, "Towards highly reliable enterprise network services via inference of multi-level dependencies," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 13–24, 2007.

[34] M. Kim, R. Sumbaly, and S. Shah, "Root cause detection in a service-oriented architecture," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1, pp. 93–104, 2013.

[35] Y. Meng, S. Zhang, Y. Sun, R. Zhang, Z. Hu, Y. Zhang, C. Jia, Z. Wang, and D. Pei, "Localizing failure root causes in a microservice through causality inference," in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*. IEEE, 2020, pp. 1–10.

[36] M. Ma, J. Xu, Y. Wang, P. Chen, Z. Zhang, and P. Wang, "Automap: Diagnose your microservice-based web applications automatically," in *Proceedings of The Web Conference 2020*, 2020, pp. 246–258.

[37] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, D. Liu, Q. Xiang, and C. He, "Latent error prediction and fault localization for microservice applications by learning from system trace logs," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 683–694.

[38] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, vol. 89. Presses universitaires de Louvain, 2015, pp. 89–94.

[39] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016.

[40] B. J. Radford, L. M. Apolonio, A. J. Trias, and J. A. Simpson, "Network traffic anomaly detection using recurrent neural networks," *arXiv preprint arXiv:1803.10769*, 2018.