KPIRoot: Efficient Monitoring Metric-based Root Cause Localization in Large-scale Cloud Systems

Wenwei Gu*, Xinying Sun‡, Jinyang Liu*, Yintong Huo*, Zhuangbin Chen†, Jianping Zhang*, Jiazhen Gu*, Yongqiang Yang‡, and Michael R. Lyu*
*The Chinese University of Hong Kong, China.
Email: {wwgu21, jyliu, ythuo, jpzhang, lyu}@cse.cuhk.edu.hk, jiazhengu@cuhk.edu.hk
†Sun Yat-sen University, China. Email: {chenzhb36}@mail.sysu.edu.cn
‡Huawei Cloud Computing Technology Co., Ltd, China. Email: {sunxinying1, yangyongqiang}@huawei.com

***Abstract*—To ensure the reliability of cloud systems, their runtime status reflecting the service quality is periodically monitored with monitoring metrics, *i.e.*, KPIs (key performance indicators). When performance issues happen, *root cause localization* pinpoints the specific KPIs that are responsible for the degradation of overall service quality, facilitating prompt problem diagnosis and resolution. To this end, existing methods generally locate root-cause KPIs by identifying the KPIs that exhibit a similar anomalous trend to the overall service performance. While straightforward, solely relying on the similarity calculation may be ineffective when dealing with cloud systems with complicated interdependent services. Recent deep learning-based methods offer improved performance by modeling these intricate dependencies. However, their high computational demand often hinders their ability to meet the efficiency requirements of industrial applications. Furthermore, their lack of interpretability further restricts their practicality. To overcome these limitations, we propose KPIRoot, an effective and efficient method for root cause localization integrating both advantages of similarity analysis and causality analysis, where similarity measures the trend alignment of KPI and causality measures the sequential order of variation of KPI. Furthermore, we leverage symbolic aggregate approximation to produce a more compact representation for each KPI, enhancing the overall analysis efficiency of the approach. The experimental results show that KPIRoot outperforms seven state-of-the-art baselines by 7.9%∼28.3%, while time cost is reduced by 56.9%. Moreover, we share our experience of deploying KPIRoot in the production environment of a large-scale cloud provider Cloud $\mathcal{H}^*$.**

***Index Terms*—Root Cause Localization, Cloud System Reliability, Cloud Monitoring Metrics, Cloud Service Systems**

I. Introduction

Large-scale cloud systems, such as Microsoft Azure, Amazon Web Services, and Google Cloud Platform, have revolutionized the computing infrastructure landscape, providing scalable, flexible, and cost-effective services to worldwide users on a 7×24 basis [1], [2]. However, the inherent complexity and scale of cloud service systems make performance issues (*e.g.*, slow application response time, service outages, and resource overload) an inevitability [3], [4], which may lead to potential violations of Service Level Agreements (SLAs), causing user dissatisfaction and financial losses [5]. Thus, promptly identifying and resolving performance issues has become a significant concern for both cloud vendors and users [6].

Cloud vendors usually collect real-time key performance indicators (KPIs) to monitor the health status of their services [7]. Anomaly detection is conducted over these KPIs to identify performance issues based on this KPI data [8]–[10]. For example, if the resource utilization rate is continuously high, it may indicate an imminent service overload and performance degradation. However, due to the scale of cloud systems, it is infeasible to analyze the KPI of each instance (*e.g.*, VM and container) individually. Since a cloud service typically consists of many instances, a common way is to monitor specific KPIs that can reflect the overall performance of the service, *e.g.*, latency, error count, and traffic, which we refer to as *alarm KPIs*. Automated performance issue detection can thus be realized through configuring alerting rules or performing anomaly detection algorithms on such alarm KPIs. These underlying KPIs of individual instances or VMs within a cloud service may not be directly analyzed due to the scale of cloud systems. However, their collective behavior significantly influences the alarm KPIs.

When a performance issue is detected (*i.e.*, the alarm KPI is abnormal), it is crucial to identify the root cause (*e.g.*, which underlying instances cause the abnormal performance of the service). However, pinpointing the root cause is a non-trivial task since the monitored alarm KPI is highly aggregated and often derived [11], *i.e.*, the correlation between the underlying KPIs and the alarm KPI is complicated and hard to understand. Even experienced software reliability engineers (SREs) can struggle to pinpoint the specific KPIs that contribute to the root cause. Such a manual approach is like finding a needle in a haystack, which is tedious and time-consuming. Hence, the automated root cause localization method is an urgent requirement for prompt performance issue resolution.

In particular, a practical root cause localization approach for KPIs from cloud systems should meet the *efficiency* and *interpretability* requirements [12]. Specifically, due to the huge volume of underlying KPIs and the tight time-to-resolve pressure, the approach needs to be able to process large amounts of data (*e.g.*, thousands of KPIs) efficiently (*e.g.*, in seconds). Furthermore, the approach should produce interpretable results to help engineers take effective remedy actions, which is essen-

Jiazhen Gu is the corresponding author.
*Due to the company policy, we anonymize the name as Cloud $\mathcal{H}$.

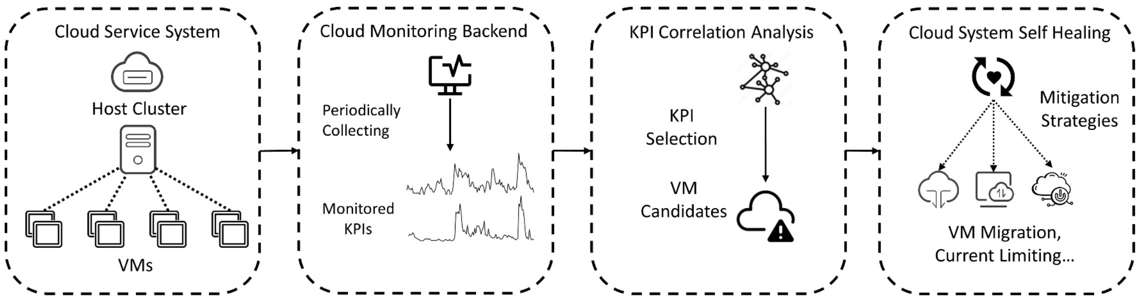

Fig. 1. The Overall Pipeline of Root Cause Localization in Cloud $\mathcal{H}$

tial in the maintenance of cloud systems. Existing root cause localization methods typically adopt statistics or deep learning models. Statistic-based methods adopt Kendall, Spearman, and Pearson correlation to compute the linear relationships between KPIs and find the root cause [13]. However, these methods have high computational costs to calculate the correlation for every KPI pair and also suffer from low accuracy in handling complicated KPIs from cloud systems [14]. Some recent studies [11] adopt deep learning models (*e.g.*, graph neural networks) to model the KPI relationships for root cause localization. However, such methods suffer from high computation costs and lack interpretability [15], [16].

To address the above limitations, we propose KPIRoot, an effective and efficient root cause localization approach to identify the root cause underlying KPIs when an anomaly in the monitored alarm KPI is detected in cloud systems. To meet the efficiency requirement, KPIRoot first adopts the Symbolic Aggregate Approximation (SAX) representation to downsample the time-series data of KPIs and facilitate extracting the anomaly segments. Through filtering out the normal KPI data, KPIRoot can focus on the anomaly patterns instead of the whole time series, which greatly optimizes efficiency. Then, KPIRoot conducts both the similarity and causality analysis to localize the root cause KPIs. Specifically, the underlying KPIs with a high similarity of anomaly patterns to the alarm KPI are more likely to trigger the alert and be the root causes. On the other hand, causality analysis is used to validate the cause and effect in the temporal dimension, *i.e.*, the anomaly pattern of root cause KPIs should happen before that of the alarm KPI. Finally, KPIRoot combines the similarity and causality analysis results to produce a correlation score for each underlying KPI. The higher the score, the more possibility there is that the KPI is the root cause. The time complexity of KPIRoot is $\mathcal{O}(\sqrt{n})$ (n is the length of the KPIs), which allows it to process thousands of KPIs in seconds, thus facilitating real-time performance issues resolution.

To evaluate the effectiveness of our proposed KPIRoot, we conducted extensive experiments based on large-scale real-world KPI data from a large cloud vendor. The experimental results demonstrate that KPI can pinpoint root cause KPIs more accurately compared with seven baselines with an F1-score of 0.850 and Hit Rate@10 of 0.917. On the other hand, KPIRoot largely reduces the computation cost with an execution time of around 5 seconds, which facilitates engineers diagnosing root causes in real-time. In particular, we have successfully deployed our approach in the cloud service system of Cloud $\mathcal{H}$ since *Nov 2022* and successfully localized the true root cause of ten performance issues of emergence level with 100 accuracy, without affecting the customer. We also share the industrial experience in practice.

We summarize the main contributions of this work as follows:

- We introduce KPIRoot, an effective and efficient method to localize the underlying KPIs that cause the anomaly. KPIRoot adopts the SAX representation for downsampling and combines both the similarity and causality of anomaly patterns of KPIs to identify the root cause. Such designs meet the practical requirements of efficiency and interpretability, making KPIRoot feasible to be deployed in large-scale cloud systems.
- Extensive experiments on three industrial datasets collected from Cloud $\mathcal{H}$'s large-scale cloud system demonstrate the effectiveness of KPIRoot, *i.e.*, 0.85 F1-score and 0.917 Hit@10 rate. The average execution time of KPIRoot is around 5 seconds, significantly outperforming seven state-of-the-art baselines.
- We have successfully deployed KPIRoot into the troubleshooting system of a large-scale cloud service system of Cloud $\mathcal{H}$ since *Nov 2022*. It has successfully analyzed ten emerging performance issues with 100 accuracy, and none of the issues affected the customer. The success stories of our deployment confirm the applicability and effectiveness of our method.

II. Background and Motivation

In this section, we briefly introduce the KPI-based root cause analysis in cloud service systems and show the root cause localization practice in a large-scale cloud service system of Cloud $\mathcal{H}$ with a real case.

A. KPI-based Root Cause Localization in Cloud Systems

Ensuring optimal performance and reliability in cloud systems is of great importance. Performance anomalies like hardware malfunctions, network overloads, and security violations can significantly influence the performance of cloud systems

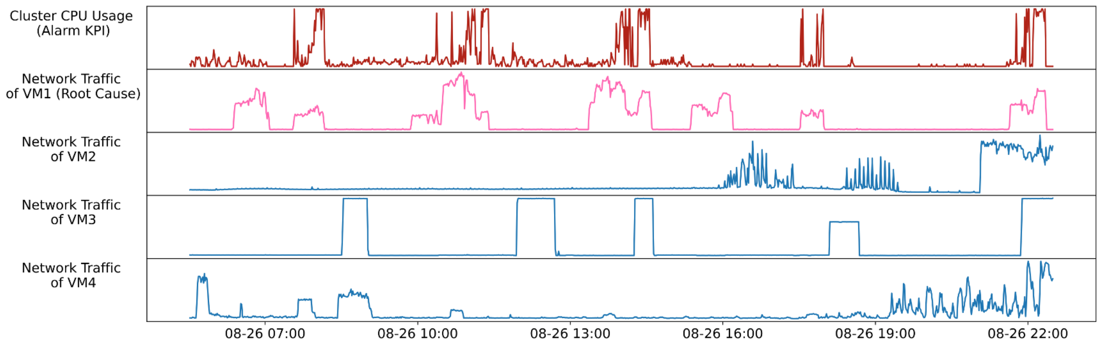

Fig. 2. An Industrial Case in Cloud $\mathcal{H}$

and violate SLA [17]. Consequently, the need for run-time status and performance monitoring of cloud systems is in demand. Key Performance Indicators (KPIs) serve as informative tools that monitor the overall status of various components of cloud systems [18], providing helpful insights that aid in the identification of potential anomalies [19], and even proactively predicting these performance issues before they escalate into catastrophic failures [20]. Some common KPIs in cloud systems include CPU usage, memory usage, network bandwidth, latency, error rates, and service QPS (queries per second).

The cloud service system has become increasingly huge in scale and produces larger volumes of monitoring data. The highly interconnected nature of cloud systems incurs problems that performance failures can spread from one component to other components. Consequently, the failure diagnosis, root cause localization, and performance debugging in large cloud systems are more complex than before [21], [22]. In real-world applications, monitoring a large number of KPIs is computationally intensive, thus a more practical way is monitoring the aggregated KPI and configuring alerts.

Specifically, in large-scale cloud service clusters, large amounts of virtual machines (VMs) operate concurrently to provide tenants with various services. A special KPI is the "*alarm KPI*" that triggers alerts when a performance issue like an overload of CPU usage in the entire cluster happens. In large-scale cloud systems, service may consist of large amounts of VMs working together to respond to cloud users' demands [23]. Given the scale of these systems, individual monitoring of each VM becomes infeasible. Instead, software reliability engineers often utilize alarm KPIs as a more effective approach to oversee the overall performance of the service. When the alarm KPI indicates abnormal activity, it becomes crucial to identify which VMs are the root causes. The root cause refers to the specific VMs that trigger the anomaly within the alarm KPI. For instance, if the alarm KPI is triggered due to a fairly high CPU usage, the root cause could be the particular VMs that directly cause the resource overload. Such a setup allows for the proactive identification of performance issues. In addition to the alarm KPI, other KPIs monitor the bytes per second (bps) and packets per second (pps) of each VM in the cluster [24]. These KPIs offer valuable insights into the data traffic of each user, serving as indicators of their workload.

The overall pipeline of root cause localization using monitoring KPI in Cloud $\mathcal{H}$ is shown in Fig.1. Cloud service providers typically have many data centers spread across different regions. Each region consists of multiple, isolated locations known as availability zones to ensure low latency and high availability [25]. Users can create their VMs in any region that best fits their needs. Then, the behavior of both the host CPU cluster and the VMs is continuously monitored and recorded through KPIs including CPU usage, memory usage, and netflow throughput. Next, KPI correlation analysis is conducted to understand the dependencies between each VM and the host cluster. Based on the KPI correlation analysis, mitigation strategies such as VM migration or throttling are enacted to alleviate the overload in the system. In our paper, we focus on the third and the most significant part, namely root cause analysis, and propose KPIRoot.

B. A Motivating Example

In a cloud system, there exist intrinsic correlations between the KPIs of individual VMs and the alarm KPI [26], which is a crucial part of RCA. Take the CPU usage in cloud systems as an example, the correlation is based on the fundamental principle of resource allocation within a cloud system that each VM is allocated a portion of the cluster's resources like CPU [27]. When a VM's workload increases, it consumes more CPU resources, thereby affecting the overall CPU usage. However, the relationship between the KPIs of individual VMs and the overall CPU usage of the cluster is complex and non-linear [28]. This complexity is due to the sophisticated architecture of modern cloud systems and the principles of resource allocation they employ. In other words, these mechanisms ensure that the resource usage of one VM does not significantly impact others, thereby preventing a single VM from monopolizing the CPU [29]. Thus, the bulge of the workload KPI of a single VM does not necessarily lead to alarm KPI trigger alerts.

To effectively identify the root cause of performance anomaly, we capture the correlations between the VM KPIs and the alarm KPI that depicts the contribution of VMs

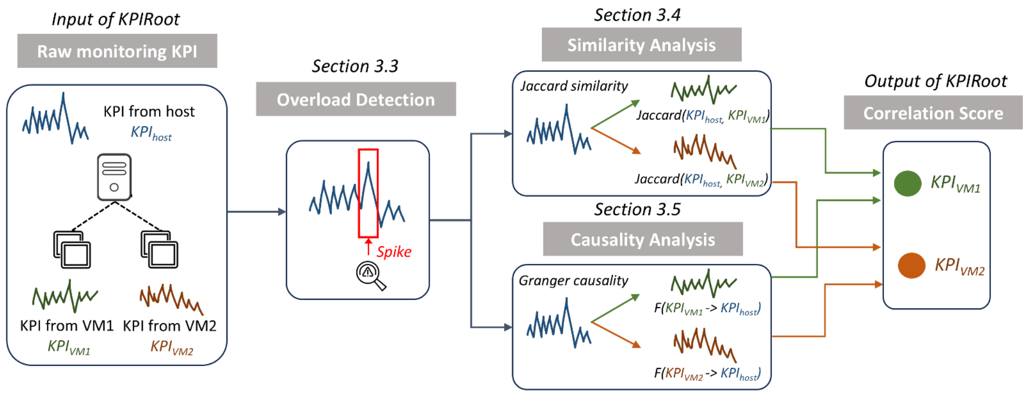

Fig. 3. The Overview of Our Proposed Method KPIRoot

to the detected performance anomaly. This correlation often manifests in a similar waveform between the VM's KPIs and the alarm KPI. For example, a sudden surge in a VM's data traffic would likely lead to an increased demand for CPU resources, which would be reflected as a spike in the KPI of the cluster's CPU usage [30]. The KPI correlation analysis approach aiming to mine the inherent correlations in KPI data can be leveraged to pinpoint the root causes of system alerts. In our case, similarity and causality analysis are adopted. Firstly, similarity analysis allows us to identify which VMs are behaving similarly to the overall system's performance, as reflected by the alarm KPI. Therefore, similarity analysis can help narrow down the potential root causes of the anomaly. Secondly, causality analysis is critical as it allows us to determine which changes in VM KPIs occurred before the anomaly, thus providing clues as to which VMs might have triggered the anomaly.

An industrial case in a real-world cloud system cluster of Cloud $\mathcal{H}$ is shown in Fig.2. There is an alarm KPI monitoring the overall CPU usage of the cluster, and several VM KPIs monitoring the network traffic of individual VMs. For the purpose of discussion, we focus on four of the VM KPIs. We can observe that the waveforms of VM2 and VM4 have weak alignments with the fluctuations in the alarm KPI, indicating a lower correlation, and thus are unlikely to be significant contributors to the CPU overload. The KPI of VM1 and VM3 exhibit a high degree of similarity to the alarm KPI, indicating they are potential root causes for the anomaly. However, to ascertain the true root cause of the CPU overload, time series causality, *i.e.*, chronological order of events should also be taken into consideration. As confirmed by the SREs, it is VM1, not VM3, which is the true root cause of the CPU overload. This is because the spike in VM1's KPI precedes the CPU overload anomaly, while the spike in VM3's KPI happens slightly after the anomaly, indicating that it is an outcome, not a cause of the anomaly. Indeed, in a cloud system, a VM's increase in resource consumption usually precedes the CPU overload due to temporal causality, which is why we take temporal causality into consideration in our method.

III. Methodology

In this section, we present KPIRoot, an automated approach for root cause localization with monitoring KPIs in cloud systems. We first formulate the problem we target. Then we provide an overview of the proposed method. Next, we elaborate on each part of our method, *i.e.*, anomaly segment detection, similarity analysis, and causality analysis. We finally analyze the complexity of our proposed algorithm.

A. Problem Formulation

The goal of our work is to identify the root causes of performance anomalies like CPU overload in large-scale cloud systems, based on the alarm KPI and observed individual KPIs. The root causes are the VMs that influence the system service quality. By throttling the throughput of these VMs, we can alleviate the system-level anomaly and restore service quality. Given the alarm KPI that monitors the status of the host cluster $X_{host} \in R^n$ and the monitored KPIs of VMs, *e.g.*, the netflow of them $X_i \in R^n, i \in \{1, 2, ..., m\}$, where N denotes the number of observations collected at an equal interval and m is the number of monitored VMs. To determine the true root cause of the detected anomaly, a correlation score $c_i \in [0, 1]$ that represents the contribution of a VM KPI to the anomaly is calculated. Then the root causes can be obtained by ranking the correlation score and KPIs with the top K scores are deemed as root causes.

B. Overview

The overview of KPIRoot is shown in Fig.3, which consists of three key components, namely, anomaly segment detection, similarity analysis, and causality analysis. Given the raw monitoring KPI, to make the RCA more efficient and meet the real-time requirement of industrial deployment, we propose to adopt SAX representation to downsample the raw KPI. Then KPIRoot detects the potential anomaly segments in the downsampled alarm KPI of the host cluster (Section.III-C). In this step, a score that describes the variation trend of KPI will be computed, an anomaly segment will automatically extracted around the spike. Then KPIRoot conducts a similarity analysis

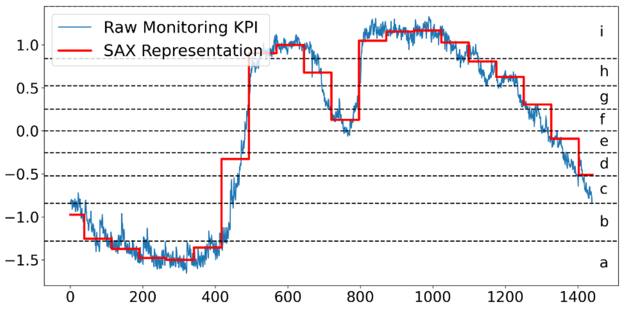

Fig. 4. An Illustration of SAX Representation

to compute the similarity between VM KPIs and the alarm KPI during the anomaly period (Section.III-D). This analysis provides insights into how each VM influences the host cluster by measuring the alignment of the KPI trends. A causality analysis is then conducted (Section.III-E) to identify the cause-and-effect between the VM KPIs and the alarm KPI. In our case, we utilize Granger causality. The results from the similarity and causality analyses are then combined to compute a correlation score for each KPI.

C. Anomaly Segment Detection

To make KPIRoot efficient and meet the industrial requirement of real-time identification, we propose to adopt Symbolic Aggregate Approximation (SAX) [31]. SAX has several advantages in KPI analysis: First, SAX allows for a significant reduction in the dimension of the raw KPI, which can make subsequent similarity computation more efficient [32]. Second, SAX can effectively filter out the noise and highlight the significant patterns in the KPIs by aggregating several consecutive data points into a single "symbol" [33]. Specifically, the raw KPI x of length n will be represented as a w-dimensional vector $P = \{p_1, p_2, ..., p_w\}$, where the j^{th} element can be calculated as follows:

$$p_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} x_j \tag{1}$$

In other words, to reduce the dimension of KPI from n to w, the KPI is divided into w equal-sized subsequences. The mean value of the subsequence is calculated and a vector of these values becomes the Piecewise Aggregate Approximation (PAA) representation [34]. Indeed, PAA representation is intuitive and simple, yet shows an approximate performance compared with more sophisticated dimension reduction representations like Fourier transforms and wavelet transforms [31]. Before converting it to the PAA, we normalize each KPI to have a mean of zero and a standard deviation of one.

In the industrial scenario, a fixed threshold method (e.g., CPU usage higher than 80%) is commonly used to detect system resource usage anomalies. However, fixed thresholds can be limiting as they do not adapt to changes in the system's behavior over time. Typically, an anomaly refers to a state where the system's resources, such as CPU, memory, or network bandwidth, are being utilized at their maximum capacity and will cause performance issues for the system. However, in a dynamic cloud system, at which threshold an anomaly occurs can shift. Specifically, during periods of low demand, a sudden spike in resource usage might be considered an anomaly. However, during peak demand periods, the system might be designed to handle much higher resource usage, thus the same usage level would not be considered an anomaly. Furthermore, the individual preferences of engineers make the setting of universally acceptable static thresholds complex. What might be a suitable threshold for one engineer could be too high or too low for another, leading to potential issues being overlooked or an excessive number of false alarms [35].

Technically, anomaly segment detection can be formulated as a spike detection problem. By detecting an uprush in workload, the early warning of potential system anomaly can be identified and root cause localization will be enabled. A score that describes the variation trend of a KPI is computed as follows:

$$r_i = \frac{\sum_{k=i}^{i+l-1} p_k}{\sum_{j=i-l}^{i-1} p_j} \tag{2}$$

where l denotes the historical lags taken into consideration. If the value r_i is greater than a large threshold γ, it suggests that the usage of resources as indicated by the KPI starts to undergo a spike and we denote the start point of overload as t_s. Once the KPI value drops below the value of t_s, it signifies that the overload ends; the endpoint of the overload is denoted as t_e. In other words, $x_{t_e} < x_{t_s}$ and $x_{t_e-1} > x_{t_s}$.

Our approach allows for the detection of anomaly segments by considering the variation trend of KPI, effectively marking the beginning and endpoint of anomaly segments. This can be a particularly beneficial preprocess step for subsequent correlation analysis.

D. Similarity Analysis

Motivated by [14], we propose to compute the similarity of the alarm KPI and VM KPIs to measure the degree of the root cause. The intuition behind this is that if a VM is responsible for triggering an overload, its KPI should exhibit a significant similarity with the host cluster's KPI, especially during periods of overload. If a VM is indeed the root cause of an overload, it is expected that its resource usage pattern would reflect the pattern of the host resource usage.

Although there exist some approaches that can be used to calculate the similarity of monitoring KPIs, such as AID [14], HALO [36], and CMMD [11], however, in real-time cloud computing systems, timely root cause localization is paramount. Traditional algorithms such as Dynamic Time Warping (DTW) might not be suitable for such scenarios due to their high time complexity, which can be prohibitive for processing large volumes of data in a real-time manner.

Thus we transform the KPIs into symbolic sequences and then compute the similarity between these sequences using the Jaccard similarity coefficient. To obtain the discrete representation with symbols, a discretization technique that will produce symbols with equal probability is desired. As proved by [31], the normalized KPIs have nearly Gaussian distributions. It's easy to pick equal-sized areas under the Gaussian distribution curve using lookup tables for the cut line coordinates, slicing the under-the-Gaussian-curve area. Suppose we have α symbols in the SAX representation, then the breakpoints refer to a sort of numbers $\beta = \{\beta_1, \beta_2, ..., \beta_\alpha\}$ such that the area under normalized Gaussian distribution curve between β_i to β_{i+1} is equal to $\frac{1}{\alpha}$. The PAA representation element in Section.III-C between β_i to β_{i+1} will be assigned with the i^{th} symbol shown as follows:

$$s_i = alphabet_l, \quad if\ \beta_l \leq p_i \leq \beta_{l+1} \tag{3}$$

where, $alphabet_i$ denotes the i^{th} symbol and s_i denotes the i^{th} element of the SAX representation S. An example of SAX representation of a monitoring KPI with $w = 20, \alpha = 9$ is shown in Fig.4.

We adopt the Jaccard similarity coefficient rather than other similarity measures because of its advantages when dealing with symbolic sequences like the SAX representation [37]. Moreover, Jaccard similarity is easy to compute and can effectively capture the similarity between two symbolic sequences regardless of their lengths. This makes it very suitable for our case, where the lengths of the symbolic sequences could vary. Then, the Jaccard similarity can be computed as follows:

$$Jaccard(S_{host}, S_i) = \frac{|S_{host} \cap S_i|}{|S_{host} \cup S_i|} \tag{4}$$

where S_{host} is the SAX representation of the host cluster's KPI and S_i is the SAX representation of individual VM KPI X_i.

E. Causality Analysis

The Symbolic Aggregate Approximation (SAX) method is effective in reducing the dimension of raw KPI, however, the computation of SAX representation-based similarity does not provide any insights into the causality between VM KPIs and alarm KPIs. As mentioned by [38], the ability of Granger causality analysis to analyze the correlation between KPIs can be a key factor for improving the accuracy of the root cause localization. By using Granger Causality in conjunction with SAX representation, we can not only analyze large quantities of time series data effectively but also gain insights into the potential causality between different KPIs. That is why we take Granger Causality [39] as a supplement.

Granger Causality is a statistical hypothesis test used to determine if one KPI is useful in forecasting another KPI [40]. For instance, if a VM KPI undergoes an uprush and causes the alarm KPI to trigger alerts, *i.e.*, the change in the VM KPI precedes the changes in the alarm KPI, then Granger causality exists from the alarm KPI to the VM KPI. It should be noted that Granger Causality is unidirectional, which means that if VM KPI Granger causes alarm KPI, it does not imply that alarm KPI Granger causes VM KPI. In our case, we are interested in understanding how VM KPIs influence the alarm KPI of the host cluster, so we focus on the Granger causality from the VM KPIs to the alarm KPI. Specifically, assuming that the two KPIs can be well described by Gaussian autoregressive processes, the autoregression (AR) of alarm KPI without and with information from VM KPI can be written as follows:

$$p^t_{alarm} = \hat{a_0} + \sum_{j=1}^{q} \hat{a_j} p^{t-j}_{alarm} + \hat{\varepsilon_t} \tag{5}$$

$$p^t_{alarm} = a_0 + \sum_{j=1}^{q} a_j p^{t-j}_{alarm} + \sum_{j=1}^{q} b_j p^{t-j}_{i} + \varepsilon_t \tag{6}$$

where the first equation uses the past values of the PAA representation of host KPI X^{host} while the second includes the past values of the PAA representation of both X^{host} and X^{vm}. Furthermore, $\hat{a_j}$ is the autoregression coefficients for X^{host}, while a_j and b_j are the autoregression coefficients for X^{host} with the contribution of both X^{host} and X^{vm}'s historical values. Both $\hat{\varepsilon_t}$ and ε_t are residual terms assumed to be Gaussian and q is model order which represents the amount of past information that will be included in the prediction of the future sample. Then we conduct the F-statistic test:

$$F_{vm \to host} = \frac{\sum_{t=t_s+q}^{t_e} (\hat{\varepsilon}^2_t - \varepsilon^2_t)/q}{\sum_{t=t_s+q}^{t_e} \varepsilon^2_t/(t_e - t_s - 2q - 1)} \tag{7}$$

where $\hat{\varepsilon}^2_t$ and ε^2_t represent the mean square error (MSE) of the AR model of host KPI without and with information from VM KPI. t_s and t_e are the start point and end point of the detected overload. The F-statistic test follows an F-distribution with q and $t_e - t_s - 2p - 1$ degrees of freedom under the null hypothesis that the VM KPI does not Granger-cause the host KPI. The calculated F-statistic can be a good indicator of the VM KPI Granger-causality to the host KPI.

After both the similarity and causality analyses are performed, KPIRoot combines these two scores to create a more comprehensive correlation score for each VM KPI. Specifically, the correlation score is a weighted sum of similarity score and causality score:

$$c_i = \lambda \times Jaccard(S_{host}, S_i) + (1 - \lambda) \times F_{vm \to host} \tag{8}$$

where c_i is the correlation score between the i^{th} VM KPI and the alarm KPI. The balance weight λ is a hyper-parameter. In our experiments, this parameter is set to be 0.9.

F. Complexity Analysis

The proposed method KPIRoot is summarized in Algorithm.1. The computation of our method mainly lies in the similarity and causality analysis. In industrial practice, $w \approx \sqrt{n}$, which means the lengths of SAX representation of KPIs are roughly $\sqrt{n}$. So, the time complexity of obtaining SAX representation is $\mathcal{O}(\sqrt{n})$. On one hand, the time complexity of Jaccard similarity is directly proportional to the KPI length, so the complexity of similarity analysis is $\mathcal{O}(\sqrt{n})$. On the other hand, the complexity of Granger causality mainly depends on

Algorithm 1 KPI Root Cause Localization

```
Input: The alarm KPI of the host $X_{alarm}$; The KPIs of VMs $X_i, i \in \{1, 2, ..., m\}$;
Output: The correlation scores of VM KPIs that correlate to the anomaly of alarm KPI $c_i$
 1: for $i = 1$; $i \leq w$; $i++$ do
 2:     $p^i_{alarm} = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} x^j_{alarm}$
 3: end for
 4: // Anomaly Segment Detection
 5: $t_s = \{t | r_t = \frac{\sum_{k=t}^{t+l-1} p^k_{alarm}}{\sum_{j=t-l}^{t-1} p^j_{alarm}} > \gamma\}$
 6: $t_e = \{min(t) | p_{t_e} < p_{t_s}$ and $p_{t_e-1} > p_{t_s}\}$
 7: $p_{alarm} = p_{alarm}[t_s : t_e]$
 8: $s^i_{alarm} = \{alphabet_l,\ s.t.\ \beta_l \leq p^i_{alarm} \leq \beta_{l+1}\}$
 9: for $i = 1$; $i \leq m$; $i++$ do
10:     // Similarity Analysis
11:     for $k = 1$; $k \leq m$; $k++$ do
12:         $p^k_i = \frac{w}{n} \sum_{j=\frac{n}{w}(k-1)+1}^{\frac{n}{w}k} x^k_i$
13:         $p_i = p_i[t_s : t_e]$
14:         $s^k_i = \{alphabet_l,\ s.t.\ \beta_l \leq p^k_i \leq \beta_{l+1}\}$
15:     end for
16:     $Jaccard(S_{host}, S_i) = \frac{|S_{host} \cap S_i|}{|S_{host} \cup S_i|}$
17:     // Causality Analysis
18:     for $t = t_s + q$; $t < t_e$; $t++$ do
19:         $p^t_{alarm} = \hat{a_0} + \sum_{j=1}^{q} \hat{a_j} p^{t-j}_{alarm} + \hat{\varepsilon}_t$
20:         $p^t_{alarm} = a_0 + \sum_{j=1}^{q} a_j p^{t-j}_{alarm} + \sum_{j=1}^{q} b_j p^{t-j}_i + \varepsilon_t$
21:     end for
22:     $F_{vm \to host} = \frac{\sum_{t=t_s+q}^{t_e} (\hat{\varepsilon}^2_t - \varepsilon^2_t)/q}{\sum_{t=t_s+q}^{t_e} \varepsilon^2_t/(t_e - t_s - 2q - 1)}$
23:     $c_i = \lambda \times Jaccard(S_{host}, S_i) + (1 - \lambda) \times F_{vm \to host}$
24: end for
25: return $c_i$
```

the autoregression of P_{host}, which is $\mathcal{O}(\sqrt{n} \times q^3)$, where q is the time lag of Granger causality (usually very small). Thus, the complexity of KPIRoot is $\mathcal{O}(\sqrt{n} \times (q^3 + 2))$. As a comparison, the time efficiency of methods like AID (based on DTW) is $\mathcal{O}(n^2)$, let alone more complex deep learning-based methods like CMMD. Therefore, KPIRoot is a more suitable method for industrial applications that demand real-time root cause localization.

IV. EVALUATION

To fully evaluate the effectiveness of our proposed approach, KPIRoot, we use three real-world monitoring KPI datasets from the cloud service systems of Cloud $\mathcal{H}$. Particularly, we aim to answer the following research questions (RQs):

- RQ1: How effective is KPIRoot compared with KPI root cause localization baselines?
- RQ2: How effective is each component of KPIRoot in root cause localization?
- RQ3: How efficient is KPIRoot in localizing root cause KPIs compared to baselines?

TABLE I
STATISTICS OF INDUSTRIAL DATASET

Industrial	Dataset A	Dataset B	Dataset C
Host Clusters	16	6	7
VM Number	120∼803	21∼26	41∼57
KPI Length	5,928,480	17,040	37,200
Root Causes	4∼36	3∼8	2∼15

A. Experiment Setting

1) Datasets: To confirm the practical significance of KPIRoot, we collect three datasets from large-scale online services in three Available Zones (AZs) of Cloud $\mathcal{H}$. The statistics of three industrial datasets are shown in Table I. Various VM KPIs and alarm KPIs monitor the status of the service. The VM KPIs typically measure the healthy status of each VM, including resource usage metrics like CPU, memory, I/O, and bandwidth usage. The alarm KPI monitors the runtime status at the host cluster level, which is usually positively correlated to the VM KPIs. Both the artifacts and data are available on https://github.com/WenweiGu/KPIRoot.

2) Evaluation Metrics: In the following experiments, the F1-score is utilized to evaluate the performance of root cause localization results. We employ Precision: $PC = \frac{TP}{TP+FP}$, Recall: $RC = \frac{TP}{TP+FN}$, F1 score: $F1 = 2 \cdot \frac{PC \cdot RC}{PC+RC}$. To be specific, TP is the number of correctly localized VM KPIs; FP is the number of incorrectly predicted VM KPIs; FN is the number of root cause VM KPIs that failed to be predicted by the model. F1 score is the harmonic mean of the precision and recall. In real-world applications, since the number of root cause KPIs is unknown, software engineers will first investigate top k recommended results by root cause localization methods. Hit Rate@k is a widely used metric to measure whether the correct root causes (in our case, the root cause VM KPIs) are within the recommended top k results. We adopt Hit Rate@5 and Hit Rate@10 as evaluation metrics in our experiments.

B. Experimental Results

*1) **RQ1** The effectiveness of KPIRoot:* To answer this research question, we compare the performance of KPIRoot with three statistical correlation measurements based methods, namely, Kendall correlation, Spearman correlation, and CloudScout [13], a DTW distance-based method AID [14], a graph centrality-based method LOUD [41], a conditional entropy-based method HALO [36] and a graph neural network based method CMMD [11]. The results are shown in Table II, where the best F1 scores, Hit@5 and Hit@10 are all marked with boldface, while the second-best results are underlined. We can see that the average F1 scores, Hit@5 and Hit@10 of KPIRoot outperform all baseline methods in three datasets. In Dataset B and Dataset C, we can observe that the improvement achieved by KPIRoot is more significant than in Dataset A. This is because Dataset B and Dataset C focus on KPIs (*e.g.*, requests rate) related to the load balancer that manages the distribution

TABLE II
EXPERIMENTAL RESULTS OF DIFFERENT ROOT CAUSE LOCALIZATION METHODS

Methods	Dataset A			Dataset B			Dataset C		
	F1 Score	Hit@5	Hit@10	F1 Score	Hit@5	Hit@10	F1 Score	Hit@5	Hit@10
Kendall	0.651	0.562	0.728	0.605	0.594	0.770	0.657	0.635	0.727
Spearman	0.681	0.587	0.753	0.619	0.591	0.737	0.681	0.598	0.715
CloudScout	0.699	0.612	0.788	0.673	0.607	0.772	0.715	0.612	0.706
LOUD	0.736	0.652	0.813	0.736	0.625	0.824	0.709	0.653	0.829
AID	0.746	0.652	0.749	0.673	0.618	0.794	0.665	0.613	0.729
HALO	0.734	0.651	0.842	0.632	0.569	0.811	0.719	0.635	0.789
CMMD	0.776	0.632	0.833	0.679	0.594	0.848	0.721	0.667	0.801
KPIRoot	**0.859**	**0.731**	**0.909**	**0.860**	**0.749**	**0.946**	**0.829**	**0.713**	**0.895**

TABLE III
EXPERIMENTAL RESULTS OF THE ABLATION STUDY OF KPIROOT

Methods	Dataset A			Dataset B			Dataset C		
	F1 Score	Hit@5	Hit@10	F1 Score	Hit@5	Hit@10	F1 Score	Hit@5	Hit@10
KPIRoot *w/o* Similarity	0.735	0.646	0.797	0.709	0.694	0.777	0.659	0.627	0.780
KPIRoot *w/o* Causality	0.801	0.694	0.858	0.748	0.706	0.869	0.731	0.675	0.823
KPIRoot	**0.859**	**0.731**	**0.909**	**0.860**	**0.749**	**0.946**	**0.829**	**0.713**	**0.895**

of network traffic across physical machines, which makes the VM request rate anomalies inherently precede host cluster anomalies. It should be noted that, as indicated in Table II, the number of root causes is often larger than 5. Therefore, not all root causes can be captured within the top 5 predictions. Given this, achieving Hit@5 scores of over 70% is significant enough, as it means our method is correctly identifying a large portion of the root causes within just the top 5 predictions. Furthermore, we observe that the F1 score and Hit@10 are high enough for industrial application, further demonstrating their effectiveness.

We can observe that baseline models like Kendall, Spearman, CloudScout, and AID have worse performance. These coefficient-based methods fundamentally measure the similarity between the shape of KPIs. However, high similarity does not necessarily imply causality because a high similarity can occur due to a shared underlying cause, rather than one KPI directly influencing another KPI. Though CMMD has the ability to capture complex, nonlinear relationships between KPIs through graph attention neural networks and achieves a Hit@10 of 0.801$\sim$0.848, it still falls short of considering the causality between VM KPIs and the host cluster KPI. HALO computes the conditional entropy between VM KPIs and the host KPI, which somehow alleviates the defect of neglecting the causality between KPIs. In contrast, our method incorporates both the similarity analysis through SAX representation similarity and causality analysis through the Granger causality test, leading to better root cause localization accuracy. The LOUD method applies graph centrality to pinpoint the root causes of issues. However, the way in which the graph is constructed can significantly impact the results. As a result, the LOUD method fails to deliver optimal performance, making it less effective in accurately identifying the root causes of problems in our context.

2) ***RQ2*** *The effectiveness of components in KPIRoot:* To answer this research question, we conducted an ablation study on KPIRoot. Particularly, we compare two baseline models removing the similarity and causality analysis part of KPIRoot to investigate the contribution of these two designs.

- *KPIRoot w/o Similarity* This baseline is a variant of KPIRoot that calculates the correlation score between KPIs merely based on the Granger causality test.
- *KPIRoot w/o Causality* This baseline removes the causality analysis part and computes the correlation score based on the SAX representation similarity.

Table III shows the performance comparison between KPIRoot and its variants. In summary, the effectiveness of KPIRoot is enhanced with the utilization of similarity analysis and causality analysis, with the former making a more significant contribution. Indeed, the variant without the Granger causality test is better than all correlation coefficient-based methods. SAX representation captures the shape and trends in the data, rather than just the raw values, which is less sensitive to noise. Thus it allows for the detection of patterns that could be missed by other methods that focus only on pointwise correlations. In contrast, Pearson, Kendall, and Spearman correlations are susceptible to noise because they perform pointwise calculations. As such, outliers within the KPIs can have a significant impact on the results of the correlation coefficients. The variants without SAX representation similarity can still yield relatively satisfactory performance because Granger causality predicts the future values of a KPI based on its own past and the past of another KPI, which makes it powerful for identifying the potential causal relationships between two KPIs. While the Granger causality test may not capture the comprehensive and complex relationships between KPIs, it is still effective for identifying potential causality thus providing valuable insights for root cause analysis.

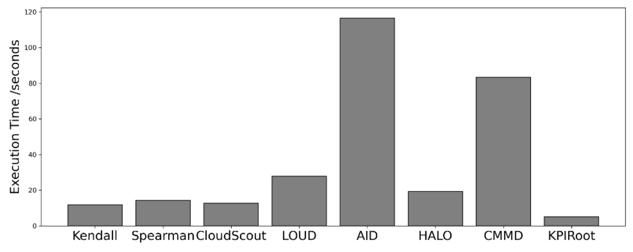

Fig. 5. Root Cause Localization Time for All Methods

3) ***RQ3*** *The efficiency of KPIRoot:* In this section, we evaluate the efficiency of KPIRoot in large-scale cloud systems of Cloud $\mathcal{H}$. The average running time of each method is shown in Fig. 5, from which we can observe that KPIRoot is the most efficient with an average execution time of around only 5 seconds, which suggests that KPIRoot is capable of providing real-time root cause analysis, meeting the requirements of large-scale cloud systems where timely identification of root causes is critical. The observed result is aligned with the time complexity analysis detailed in Section III-F. As for methods like AID and CMMD, their performances are less than satisfactory due to their inherent computational complexities. AID, with its time complexity of $\mathcal{O}(n^2)$, suffers from an average runtime of more than one hundred seconds. On the other hand, CMMD, which applies graph attention neural networks, requires high computational resources, which also leads to a slower execution time and makes it less efficient. Therefore, both AID and CMMD fail to deliver the desired levels of efficiency, particularly in large-scale, real-time environments. Baseline methods like Kendall and Spearman may seem appealing due to their lower computation times. However, these apparent gains are offset by their inferior accuracy levels. As a result, their use can lead to inaccurate root-cause diagnoses and subsequently ineffective problem-solving solutions.

In summary, the evaluation results highlight KPIRoot's superiority in terms of both efficiency and accuracy, which makes KPIRoot a highly promising tool for conducting real-time root cause analysis within large-scale cloud systems.

V. Industrial Experience

In this section, we share our experience of deploying KPIRoot in the cloud system of Cloud $\mathcal{H}$, a full-stack cloud system that consists of an infrastructure layer, a platform layer, and an application layer. To support a large number of customers, each of our services is supported by multiple clusters with tens of hundreds of virtual instances (*e.g.*, virtual router) or devices. The collective workload of each cluster is continuously monitored using an alarm KPI. When abnormal traffic impacts these services, for instance, due to overwhelming requests overloading a service, an anomaly is swiftly detected based on the alarm KPI. This triggers a root cause analysis procedure to pinpoint the specific nodes (e.g., VMs) and take prompt mitigating actions. In our previous practice, manual inspection is feasible given the limited scale of each cluster. So, we can check each specific KPI of the node, compare it with the alarm KPI (with similarity comparison tools), and find the root cause. However, this process proved to be error-prone and labor-intensive, particularly as the scale of each service expanded. On average, it took between thirty minutes to one hour to identify and mitigate the root causes.

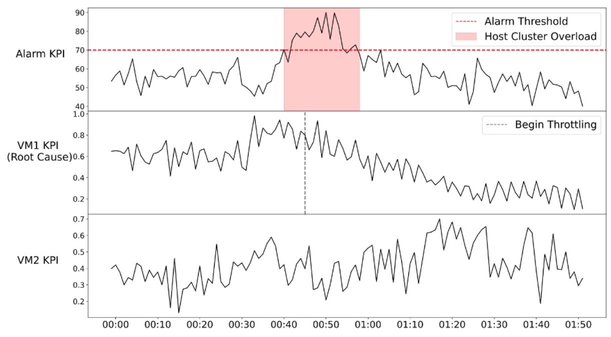

Fig. 6. Case Study of KPIRoot

To alleviate these issues, we have deployed KPIRoot in Cloud $\mathcal{H}$ since *Nov 2022*. Specifically, KPIRoot operates by automatically fetching KPIs collected from the monitoring backends and applying the algorithm to calculate the correlation score in real time. Using KPIRoot, the potential root causes are returned to engineers. In addition, visualization tools are provided, making it easier for engineers to understand the system's behavior and performance.

In Fig. 6, we demonstrate the practical application of the root cause analysis tool KPIRoot in real industrial scenarios. While our system may encompass tens to hundreds of KPIs, as outlined in Table I, for the sake of conciseness, we showcase only two KPIs in this illustration. In this case, we initially received an alert indicating that the overall traffic for the host cluster had abruptly surpassed the predefined threshold. This requires immediate measures to pinpoint the root cause and throttle its throughput to avoid resource exhaustion within the cluster. However, this is quite challenging given the large number of KPIs needed to check, and the root-cause KPI may not be readily identifiable visually, as its shape similarity may not correspond directly with the alarm KPI. Given that, the root cause analysis takes tens of minutes to one hour to check manually, leading to delayed mitigation of the sudden traffic spike. With KPIRoot, the root cause of KPI can be quickly localized, generally within five minutes. With this result, we throttle the throughput of VM1 immediately after the alarm KPI is fired. As shown in Figure 6, the overall traffic is limited, and the alarm KPI returns back to a normal range quickly.

KPIRoot has been deployed in all major regions of our company, covering eighteen critical network services, *e.g.*, Linux Virtual Server (LVS), NGINX, Network Address Translation (NAT), and DNS services. It has been serving in our production environment for more than ten months, reducing the average root cause localization time from 30 minutes to 5 minutes. Following the deployment of the KPIRoot service, the feedback from engineers has been overwhelmingly positive. In terms of computational efficiency, KPIRoot has reduced the computational load significantly compared to previous methods. The system can perform real-time RCA, identifying potential issues quickly and allowing engineers to take immediate action. In terms of accuracy, KPIRoot's design of combining similarity and causality analysis has proven highly precise in identifying root causes. This leads to more effective problem resolution and significantly reduced revenue loss.

VI. Discussion

In this section, we discuss the difference between our approach and existing root cause analysis approaches for microservice systems and why they are not applicable in our industrial scenario. Besides, we discuss the influence of SAX representation in KPIRoot. Finally, we identified some potential threats to the validity of our study.

A. Root Cause Analysis for Microservice System

Our objective shares some similarities with root cause analysis in microservice systems, however, there are several main differences in terms of the application scenarios. Firstly, rather than localizing the root causes of application/service failures in microservice systems, where these applications are at the same level, our problem is top-down root localization. When we observe an anomaly at the system level, we investigate and analyze the underlying VM instance-level information. Secondly, due to VM isolation, each VM instance operates independently and is isolated from other VMs and the host system. This leads to sparse or even non-existent invocation dependency among them, making the construction of a service dependency graph as done in existing works very challenging.

Existing Methods like FRL-MFPG [42] and ServiceRank [43] rely on the construction of a service dependency graph and the execution of a second-order random walk, which can become highly time-consuming with complexity exceeding $O(n^2)$. As for HRLHF [44], the large graph size makes causal discovery computationally intensive. Furthermore, the delay incurred by waiting for engineers to provide human feedback poses an additional obstacle for real-time localization. However, the analysis delay should be less than the sampling interval, *e.g.*, 1 minute in our practical scenarios, making these methods unsuitable for industrial deployment.

B. The Influence of SAX Representation

The anomaly segment detection serves as a precursor to root cause localization and may influence the subsequent task. Since SAX representation is a downsampling method, it may induce the omission of the original KPI information. However, the alarm KPI is carefully selected by experienced engineers based on their experience. These KPIs typically exhibit distinct anomaly patterns when system-level issues occur, making them highly reliable system health indicators, even if SAX may cause some content of information loss. Given the detected anomalies in alarm KPIs, our target is to find the VMs causing performance anomalies, which should also have obvious patterns in the VM KPI. So the SAX representation has little influence on the localization part.

C. Threats to Validity

We have identified the following potential threats to the validity of our study:

Internal threats. The implementation of baselines and parameter settings constitutes one of the internal threats to our work's validity. To mitigate these threats, we utilized the open-sourced code released by the authors of the papers or packages on GitHub for all baselines. As for our proposed approach, the source code has been reviewed meticulously by the authors, as well as several experienced software engineers, to minimize the risk of errors and increase the overall confidence in our results. For parameter settings, as our algorithm KPIRoot has few parameters, we find the most suitable configurations based on the best results obtained in different parameters.

External threats. Our experiments are conducted based on real-world datasets collected from Cloud $\mathcal{H}$ over more than two years. The evaluation requires engineers to inspect and label the root cause KPIs manually. Label noises are inevitable during the manual labeling process. However, alleviation strategies taken by engineers further ensure the accuracy of labeled root causes. Therefore, we believe the amount of noise is small and does not have a significant impact on the experiment results. On the other hand, the results may vary between different cloud service providers, industries, or specific use cases. Nevertheless, we believe that our experimental results, obtained from large-scale online systems within a prominent cloud service company serving millions of users, can demonstrate the generality and effectiveness of our proposed approach, KPIRoot.

VII. Related Work

A. Anomaly Detection in Cloud Systems

Ensuring the optimal performance of cloud systems is an imperative task. Monitoring KPIs are used to perceive the status of the cloud systems and facilitate analysis when performance anomalies occur. Many works [1], [45]–[49] have been proposed for proactively discovering the unexpected or anomalous behaviors of the multivariate monitoring metric. Anomaly detection in cloud systems has been an important and widely studied topic as it ensures the reliability and efficiency of cloud systems. However, anomaly detection is regarded as a black box module that only predicts whether an anomaly happens, which is not enough for engineers to troubleshoot the system failure. In other words, once a performance anomaly has been detected in a cloud service system, further analyses should be enacted to pinpoint some abnormal metrics that are likely to be the possible root causes of that performance anomaly.

B. Root Cause Localization in Cloud Systems

Determining the root cause of performance anomalies for online service systems has been a hot topic. The goal of root cause localization with monitoring metrics data in cloud systems is to localize a subset of the monitored KPIs. Then, they can troubleshoot these specific parts of the system to alleviate the performance anomaly. LOUD [41] assumes that the services of anomalous KPIs are likely to result in anomalous behavior of services it correlates with. Thus, LOUD applies graph centrality to identify the degree of the KPIs that correlate to the observed performance anomaly. AID [14] is an approach that measures the intensity of dependencies between monitoring KPIs of cloud services. It calculates the similarities between the status KPIs of the caller and the callee. Then, AID aggregates the similarities to produce a unified value as the intensity of the dependency. It can also be deployed as a root cause localization tool as it can output the similarity between monitoring metrics and the KPI that triggers alerts. Similarly, CloudScout [13] employs the Pearson Correlation Coefficient over KPIs at the physical machine level, such as CPU usage, to calculate the similarity between services.

There are also many works focusing on searching fault-indicating attribute combinations of KPI data. CMMD [11] is proposed to perform cross-metric root cause localization through a graph attention network to model the relationship between fundamental and derived metrics. While HALO [36] proposed a hierarchical search approach to capture the relationship among attributes based on conditional entropy and locate the fault-indicating combination. Another approach iDice [50] treats the root cause as a combination of attribute values, *i.e.*, the anomaly can be easily identified through the co-occurrence of some specific attribute dimensions. A Fisher distance-based score function is utilized for ranking the combination of the attributes, and effective combinations will be output. However, iDice is not suitable for large-scale issue reports with high-dimensional metrics from cloud systems. MID [51] employs a meta-heuristic search that automatically detects dynamic emerging issues from large-scale issue reports with higher efficiency.

It is worth noting that, in our case, the monitoring metrics are not aggregated along different attribute dimensions through complex calculations of the raw data. Indeed, the monitoring metrics in our scenario directly reflect the run-time state of an entity, *e.g.*, the throughput of a client VM. In our practice, obtaining the root cause at a granularity of metric level is enough for engineers to troubleshoot the performance anomalies. Thus, we formulate our problem as localizing a subset of the monitored KPIs.

VIII. CONCLUSION

In this paper, we propose KPIRoot, an effective and efficient framework for root cause analysis in practical cloud systems with monitoring KPIs. Specifically, KPIRoot integrates the strength of similarity analysis and causality analysis, offering a more comprehensive correlation evaluation of KPI, thus enhancing the accuracy of root cause localization. Additionally, the utilization of SAX representation of KPI significantly improves the efficiency of the method. Extensive experiments on three industrial datasets show that KPIRoot achieves 0.850 F1-Score and 0.916 Hit@10 with the highest efficiency, outperforming all the baselines. Moreover, the successful deployment of our approach in large-scale industrial applications further demonstrates its practicality.

IX. ACKNOWLEDGMENTS

The work described in this paper was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14206921 of the General Research Fund) and Fundamental Research Funds for the Central Universities, Sun Yat-sen University (No. 76250-31610005).

REFERENCES

[1] Q. Lin, K. Hsieh, Y. Dang, H. Zhang, K. Sui, Y. Xu, J.-G. Lou, C. Li, Y. Wu, R. Yao *et al.*, "Predicting node failure in cloud service systems," in *Proceedings of the 2018 26th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*, 2018, pp. 480–490.

[2] B. Yu, J. Yao, Q. Fu, Z. Zhong, H. Xie, Y. Wu, Y. Ma, and P. He, "Deep learning or classical machine learning? an empirical study on log-based anomaly detection," in *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, 2024, pp. 1–13.

[3] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *Proceedings of the 38th International Conference on Software Engineering Companion*, 2016, pp. 102–111.

[4] J. Kuang, J. Liu, J. Huang, R. Zhong, J. Gu, L. Yu, R. Tan, Z. Yang, and M. R. Lyu, "Knowledge-aware alert aggregation in large-scale cloud systems: a hybrid approach," in *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*, 2024, pp. 369–380.

[5] J. Liu, S. He, Z. Chen, L. Li, Y. Kang, X. Zhang, P. He, H. Zhang, Q. Lin, Z. Xu *et al.*, "Incident-aware duplicate ticket aggregation for cloud systems," *arXiv preprint arXiv:2302.09520*, 2023.

[6] J. Liu, S. Wang, A. Zhou, S. A. Kumar, F. Yang, and R. Buyya, "Using proactive fault-tolerance approach to enhance cloud service reliability," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 1191–1202, 2016.

[7] 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.

[8] N. Zhao, J. Zhu, R. Liu, D. Liu, M. Zhang, and D. Pei, "Label-less: A semi-automatic labelling tool for kpi anomalies," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1882–1890.

[9] N. Zhao, J. Chen, Z. Wang, X. Peng, G. Wang, Y. Wu, F. Zhou, Z. Feng, X. Nie, W. Zhang *et al.*, "Real-time incident prediction for online service systems," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 315–326.

[10] T. Huang, P. Chen, J. Zhang, R. Li, and R. Wang, "A transferable time series forecasting service using deep transformer model for online systems," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–12.

[11] S. Yan, C. Shan, W. Yang, B. Xu, D. Li, L. Qiu, J. Tong, and Q. Zhang, "Cmmd: Cross-metric multi-dimensional root cause analysis," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 4310–4320.

[12] G. Yu, P. Chen, Y. Li, H. Chen, X. Li, and Z. Zheng, "Nezha: Interpretable fine-grained root causes analysis for microservices on multi-modal observability data," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 553–565.

[13] J. Yin, X. Zhao, Y. Tang, C. Zhi, Z. Chen, and Z. Wu, "Cloudscout: A non-intrusive approach to service dependency discovery," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 5, pp. 1271–1284, 2016.

[14] T. Yang, J. Shen, Y. Su, X. Ling, Y. Yang, and M. R. Lyu, "Aid: efficient prediction of aggregated intensity of dependency in large-scale cloud systems," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 653–665.

[15] J. Zhang, W. Wu, J.-t. Huang, Y. Huang, W. Wang, Y. Su, and M. R. Lyu, "Improving adversarial transferability via neuron attribution-based attacks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 993–15 002.

[16] J. Zhang, W. Gu, Y. Huang, Z. Jiang, W. Wu, and M. R. Lyu, "Curvature-invariant adversarial attacks for 3d point clouds," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 7, 2024, pp. 7142–7150.

[17] Y. Sharma, D. Bhamare, N. Sastry, B. Javadi, and R. Buyya, "Sla management in intent-driven service management systems: A taxonomy and future directions," *ACM Computing Surveys*, 2023.

[18] Q. Cheng, D. Sahoo, A. Saha, W. Yang, C. Liu, G. Woo, M. Singh, S. Saverese, and S. C. Hoi, "Ai for it operations (aiops) on cloud platforms: Reviews, opportunities and challenges," *arXiv preprint arXiv:2304.04661*, 2023.

[19] S. Singh, R. Batheri, and J. Dias, "Predictive analytics: How to improve availability of manufacturing equipment in automotive firms," *IEEE Engineering Management Review*, 2023.

[20] S. Tuli, S. S. Gill, P. Garraghan, R. Buyya, G. Casale, and N. Jennings, "Start: Straggler prediction and mitigation for cloud computing environments using encoder lstm networks," *IEEE Transactions on Services Computing*, 2021.

[21] H. Wang, P. Nguyen, J. Li, S. Kopru, G. Zhang, S. Katariya, and S. Ben-Romdhane, "Grano: Interactive graph-based root cause analysis for cloud-native distributed data platform," *Proceedings of the VLDB Endowment*, vol. 12, no. 12, pp. 1942–1945, 2019.

[22] J. Qiu, Q. Du, K. Yin, S.-L. Zhang, and C. Qian, "A causality mining and knowledge graph based method of root cause diagnosis for performance anomaly in cloud applications," *Applied Sciences*, vol. 10, no. 6, p. 2166, 2020.

[23] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *2010 24th IEEE international conference on advanced information networking and applications*. IEEE, 2010, pp. 446–452.

[24] M. Latah and L. Toker, "Artificial intelligence enabled software-defined networking: a comprehensive overview," *IET networks*, vol. 8, no. 2, pp. 79–99, 2019.

[25] P. Kaushik, A. M. Rao, D. P. Singh, S. Vashisht, and S. Gupta, "Cloud computing and comparison based on service and performance between amazon aws, microsoft azure, and google cloud," in *2021 International Conference on Technological Advancements and Innovations (ICTAI)*. IEEE, 2021, pp. 268–273.

[26] W. Gu, J. Liu, Z. Chen, J. Zhang, Y. Su, J. Gu, C. Feng, Z. Yang, and M. Lyu, "Performance issue identification in cloud systems with relational-temporal anomaly detection," *arXiv preprint arXiv:2307.10869*, 2023.

[27] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 153–167.

[28] D. Wang, Z. Chen, J. Ni, L. Tong, Z. Wang, Y. Fu, and H. Chen, "Hierarchical graph neural networks for causal discovery and root cause localization," *arXiv preprint arXiv:2302.01987*, 2023.

[29] F. Zhou, M. Goel, P. Desnoyers, and R. Sundaram, "Scheduler vulnerabilities and coordinated attacks in cloud computing," *Journal of Computer Security*, vol. 21, no. 4, pp. 533–559, 2013.

[30] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[31] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 2003, pp. 2–11.

[32] D. Minnen, C. Isbell, I. Essa, and T. Starner, "Detecting subdimensional motifs: An efficient algorithm for generalized multivariate pattern discovery," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE, 2007, pp. 601–606.

[33] P. Senin and S. Malinchik, "Sax-vsm: Interpretable time series classification using sax and vector space model," in *2013 IEEE 13th international conference on data mining*. IEEE, 2013, pp. 1175–1180.

[34] C. Guo, H. Li, and D. Pan, "An improved piecewise aggregate approximation based on statistical features for time series mining," in *Knowledge Science, Engineering and Management: 4th International Conference, KSEM 2010, Belfast, Northern Ireland, UK, September 1-3, 2010. Proceedings 4*. Springer, 2010, pp. 234–244.

[35] N. Zhao, J. Chen, X. Peng, H. Wang, X. Wu, Y. Zhang, Z. Chen, X. Zheng, X. Nie, G. Wang *et al.*, "Understanding and handling alert storm for online service systems," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice*, 2020, pp. 162–171.

[36] X. Zhang, C. Du, Y. Li, Y. Xu, H. Zhang, S. Qin, Z. Li, Q. Lin, Y. Dang, A. Zhou *et al.*, "Halo: Hierarchy-aware fault localization for cloud systems," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 3948–3958.

[37] X. He, C. Shao, and Y. Xiong, "A non-parametric symbolic approximate representation for long time series," *Pattern Analysis and Applications*, vol. 19, pp. 111–127, 2016.

[38] L. Mariani, M. Pezzè, O. Riganelli, and R. Xin, "Predicting failures in multi-tier distributed systems," *Journal of Systems and Software*, vol. 161, p. 110464, 2020.

[39] A. Shojaie and E. B. Fox, "Granger causality: A review and recent advances," *Annual Review of Statistics and Its Application*, vol. 9, pp. 289–319, 2022.

[40] A. Arnold, Y. Liu, and N. Abe, "Temporal causal modeling with graphical granger methods," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007, pp. 66–75.

[41] L. Mariani, C. Monni, M. Pezzé, O. Riganelli, and R. Xin, "Localizing faults in cloud systems," in *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2018, pp. 262–273.

[42] Y. Chen, D. Xu, N. Chen, and X. Wu, "Frl-mfpg: Propagation-aware fault root cause location for microservice intelligent operation and maintenance," *Information and Software Technology*, vol. 153, p. 107083, 2023.

[43] M. Ma, W. Lin, D. Pan, and P. Wang, "Servicerank: Root cause identification of anomaly in large-scale microservice architectures," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 3087–3100, 2021.

[44] L. Wang, C. Zhang, R. Ding, Y. Xu, Q. Chen, W. Zou, Q. Chen, M. Zhang, X. Gao, H. Fan *et al.*, "Root cause analysis for microservice systems via hierarchical reinforcement learning from human feedback," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 5116–5125.

[45] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International conference on learning representations*, 2018.

[46] Z. Chen, J. Liu, Y. Su, H. Zhang, X. Ling, Y. Yang, and M. R. Lyu, "Adaptive performance anomaly detection for online service systems via pattern sketching," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 61–72.

[47] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 187–196.

[48] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang, "Time-series anomaly detection service at microsoft," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 3009–3017.

[49] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2828–2837.

[50] Q. Lin, J.-G. Lou, H. Zhang, and D. Zhang, "idice: problem identification for emerging issues," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 214–224.

[51] J. Gu, C. Luo, S. Qin, B. Qiao, Q. Lin, H. Zhang, Z. Li, Y. Dang, S. Cai, W. Wu *et al.*, "Efficient incident identification from multi-dimensional issue reports via meta-heuristic search," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 292–303.