Empirical Software Engineering (2026) 31:28
https://doi.org/10.1007/s10664-025-10769-0

KPIRoot+: An efficient integrated framework for anomaly detection and root cause analysis in large-scale cloud systems

Wenwei Gu[1] · Renyi Zhong[1] · Guangba Yu[1] · Xinying Sun[4] · Jinyang Liu[1] · Yintong Huo[3] · Zhuangbin Chen[2] · Jianping Zhang[1] · Jiazhen Gu[1] · Yongqiang Yang[4] · Michael R. Lyu[1]

Received: 17 February 2025 / Accepted: 3 November 2025 / Published online: 5 December 2025

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, an effective and efficient root cause localization method, KPIRoot, is proposed. It integrates 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, it leverages symbolic aggregate approximation to produce a more compact representation for each KPI, enhancing the overall analysis efficiency of the approach. However, during the deployment of KPIRoot in cloud systems of a large-scale cloud system vendor, Cloud $\mathcal{H}$. We identified two additional drawbacks of KPIRoot: 1. The threshold-based anomaly detection method is insufficient for capturing all types of performance anomalies; 2. The SAX representation cannot capture intricate variation trends, which causes suboptimal root cause localization results. We propose KPIRoot+ to address the above drawbacks. The experimental results show that KPIRoot+ outperforms eight state-of-the-art baselines by 2.9%∼35.7%, while time cost is reduced by 34.7%. Moreover, we share our experience of deploying KPIRoot in the production environment of a large-scale cloud provider Cloud $\mathcal{H}$.

Communicated by: Lei Ma, Roberto Pietrantuono.

Due to the company policy, we anonymize the name as Cloud $\mathcal{H}$

Extended author information available on the last page of the article

Keywords Root Cause Localization · Cloud System Reliability · Cloud Monitoring Metrics · Cloud Service Systems

1 Introduction

Large-scale cloud systems have become the backbone of modern computing infrastructure, offering unprecedented scalability and flexibility. Cloud platforms such as Microsoft Azure, Amazon Web Services, and Google Cloud Platform provide cost-effective services to users worldwide on a 7 × 24 basis (Lin et al. 2018; Yu et al. 2024). However, the inherent complexity and scale of these systems inevitably lead to performance issues, including slow application response times, network latency spikes, and resource contention (Lin et al. 2016b; Kuang et al. 2024). These issues can result in violations of Service Level Agreements (SLAs), causing user dissatisfaction and financial losses for both service providers and consumers (Liu et al. 2023). Consequently, the prompt identification and resolution of performance issues have become critical concerns for cloud vendors and users alike (Liu et al. 2016). Addressing these challenges is essential for maintaining the reliability and efficiency of cloud services in an increasingly digital world.

Cloud vendors usually collect real-time key performance indicators (KPIs) to monitor the health status of their services (Su et al. 2019b). Anomaly detection is conducted over these KPIs to identify performance issues based on this KPI data (Zhao et al. 2019, 2020b; Huang et al. 2022). 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 (Soldani and Brogi 2022) (*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 (Yan et al. 2022), *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 (Yu et al. 2023). 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 essential 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 (Yin et al. 2016). 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 (Yang et al. 2021). Some recent studies (Yan et al. 2022) 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 (Zhang et al. 2022, 2024).

To address the above limitations, a root cause localization framework, KPIRoot (Gu et al. 2024b), is proposed 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. By filtering out the normal KPI data, KPIRoot can focus on anomaly patterns instead of the whole time series, which optimizes efficiency. Then, KPIRoot conducts both the similarity and causality analysis to localize the root cause KPIs. Specifically, 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 likely 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 the resolution of real-time performance issues.

However, we identified several drawbacks of KPIRoot. Firstly, the threshold-based anomaly detection method employed by KPIRoot, while effective in identifying trend anomalies, struggles to detect seasonal and point anomalies. This limitation is particularly highlighted in performance issues reflected in KPIs, where seasonal fluctuations and sudden spikes or drops are common and critical to accurate anomaly detection. Secondly, although the SAX representation utilized in KPIRoot enhances the efficiency of root cause localization by downsampling the KPIs, it may not effectively capture intricate variation trends. This limitation arises from its reliance on segment averages, which can obscure variation trend details in the data essential for accurate root cause localization.

This paper extends our preliminary work, which appears as a research paper of ISSRE 2024 (Gu et al. 2024b). In particular, we extend our preliminary work in the following directions:

- We propose KPIRoot+ , an extended version of the KPIRoot framework introduced in our preliminary work (Gu et al. 2024b). There are two major differences in KPIRoot+ compared to KPIRoot. Firstly, anomaly detection is positioned as a critical precursor to root cause localization. We reveal that the original KPIRoot framework struggles to detect all types of anomalies, which are pivotal for accurate root cause localization in some cases. To address this deficiency, we have implemented a time series decomposition-based method. By supplementing the original approach based on trend variation with time series decomposition and a U-Net autoencoder, KPIRoot+ significantly improves the accuracy of anomaly detection, thus improving the subsequent root cause analysis phase. Secondly, the original Symbolic Aggregate Approximation (SAX) representation employed in KPIRoot falls short of effectively capturing intricate trends and variations due to its dependence on segment averages. This can obscure critical behavioral patterns. To overcome these limitations, KPIRoot+ incorporates an Improved SAX representation (ISAX) that further incorporates trend variation indicators. Our experiments show that KPIRoot+ performs better in terms of root cause localization accuracy but requires a similar execution time when compared with KPIRoot.

- We conduct a comprehensive evaluation of anomaly detection performance across different models, an aspect that was overlooked in KPIRoot.
- We strengthen our experimental part by including NDCG in our evaluation metrics, specifically NDCG@5 and NDCG@10. This metric measures how easily engineers can find the culprit VMs, which is crucial in our scenarios, as the most relevant root causes are prioritized for investigation.
- We conduct a sensitivity analysis on the parameters used in KPIRoot+ . The results demonstrate that our approach maintains robustness within a reasonable interval of parameter values.
- During the deployment of KPIRoot in our Cloud $\mathcal{H}$, we identified several failure cases that highlighted its limitations. We share our industrial experiences and insights on how KPIRoot+ addresses these issues.

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 KPIRoot+ can pinpoint root cause KPIs more accurately compared with seven baselines with an F1-score of 0.882 and Hit Rate@10 of 0.946. On the other hand, KPIRoot+ largely reduces the computation cost with an execution time of around 8 seconds, which facilitates engineers' diagnosis of root causes in real time. In particular, we have successfully deployed our approach in the cloud service system of Cloud $\mathcal{H}$ since *Aug 2023* and successfully localized the true root cause of ten performance issues of emergence level with 100% accuracy without affecting the customer. We also share industrial experience in practice.

We summarize the main contributions of this work, which form a superset of those in our preliminary study, as follows:

- We introduce KPIRoot+ , an effective and efficient method to localize the underlying KPIs that cause the anomaly, which is an improved version of KPIRoot. KPIRoot+ adopts the Improved 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 deploy in large-scale cloud systems. We further strengthen the anomaly detection part to make it effective for different anomaly types.
- Extensive experiments on three industrial datasets collected from Cloud $\mathcal{H}$'s large-scale cloud system demonstrate the effectiveness of KPIRoot+ , *i.e.*, 0.882 F1-score and 0.946 Hit@10 rate. The average execution time of KPIRoot+ is around 8 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.

2 Background and Motivation

In this section, we present a comprehensive overview of KPI-based root cause analysis in cloud service systems and demonstrate its practical application through a case study of root cause localization in Cloud $\mathcal{H}$, a large-scale production cloud environment.

2.1 KPI-based Root Cause Localization in Cloud Systems

Ensuring 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 and violate SLA (Sharma et al. 2023). 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 (Cheng et al. 2023), providing helpful insights that aid in the identification of potential anomalies (Singh et al. 2023), and even proactively predicting these performance issues before they escalate into catastrophic failures (Tuli et al. 2021). 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 causes problems, such as performance failures, which can spread from one component to another component. Consequently, the failure diagnosis, root cause localization, and performance debugging in large cloud systems are more complex than before (Wang et al. 2019; Qiu et al. 2020). In real-world applications, monitoring a large number of KPIs is computationally intensive. Thus, a more practical way is to monitor the aggregated KPI and configure 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 (Wickremasinghe et al. 2010). 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. Therefore, while our analysis focuses on identifying the most indicative KPIs, the term "root cause" in this paper consistently refers to the VM instance from which these KPIs originate. 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 (Latah and Toker 2019). These KPIs offer valuable insights into the data traffic of each user, serving as indicators of their workload.

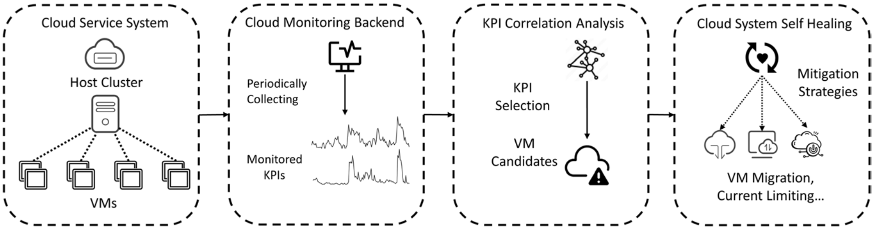


Fig. 1 The Overall Pipeline of Root Cause Localization in Cloud $\mathcal{H}$

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 (Kaushik et al. 2021). 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 system overload. In our paper, we focus on the third and most significant part, namely root cause analysis, and propose KPIRoot+ .

2.2 A Motivating Example

In a cloud system, there exist intrinsic correlations between the KPIs of individual VMs and the alarm KPI (Gu et al. 2024a), 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 (Cortez et al. 2017). 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 (Wang et al. 2023a). 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 (Zhou et al. 2013). 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 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 (Beloglazov and Buyya 2012). 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 behav-

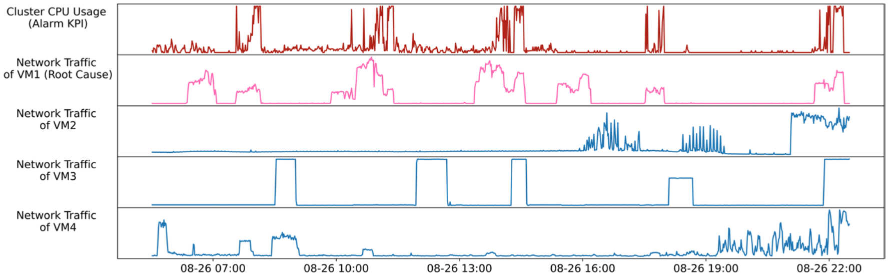


Fig. 2 An Industrial Case in Cloud $\mathcal{H}$

ing 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.

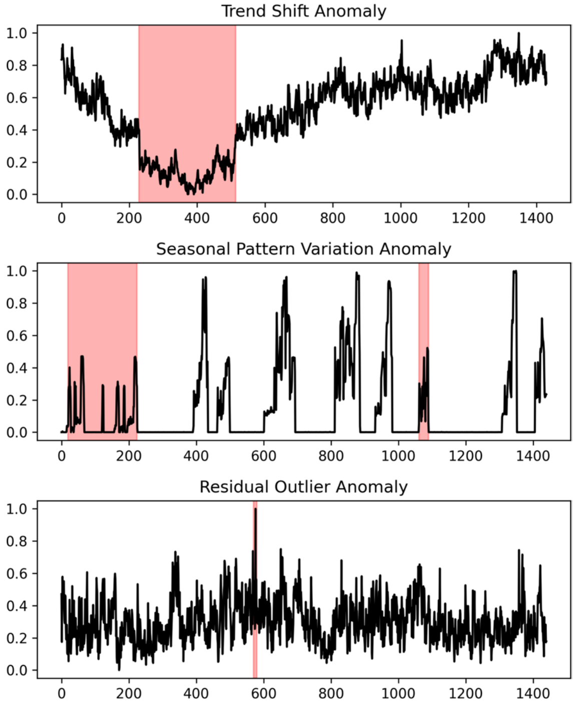
An industrial case in a real-world cloud system cluster of Cloud $\mathcal{H}$ is shown in Fig. 2. There is an alarm KPI monitoring the overall CPU usage of the cluster, and several VM KPIs monitor the network traffic of individual VMs. For the purpose of the discussion, we will focus on four of the 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 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.

2.3 Different Types of Performance Anomalies

Our previous work KPIRoot (Gu et al. 2024b) predominantly focuses on detecting trend anomalies using a threshold-based method. While effective for identifying gradual or sustained shifts in performance metrics, this approach may not adequately capture the breadth of anomalies that can occur in Cloud $\mathcal{H}$. Specifically, seasonal and residual anomalies, which manifest as periodic deviations or abrupt, unexpected changes, respectively, might not be sufficiently detected by a threshold method alone.

In Fig. 3, we observe three distinct types of performance anomalies across different monitoring metrics within Cloud $\mathcal{H}$. The first anomaly is a trend anomaly characterized by a sudden downward shift in throughput on a network interface card (NIC). This abrupt change can be indicative of packet loss, which might occur due to network congestion, hardware malfunctions, or configuration errors. The second case illustrates seasonal anomalies in NIC throughput, with unexpected deviations occurring in the area marked with red spans. The

Fig. 3 Different Anomaly Types in Cloud $\mathcal{H}$



anomaly could suggest issues like batch jobs running at non-standard times or misconfigured scheduling that leads to a throughput drop. The third example presents a residual anomaly in the average throughput on another NIC. Such short-duration spikes are neither part of a long-term trend nor follow a seasonal pattern, hinting at sporadic issues such as brief network outages, hardware failures, or security incidents like DDoS attacks. All these three types of performance anomalies can have severe impacts on service performance and reliability.

3 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.*, time series decomposition-based anomaly segment detection, similarity analysis, and causality analysis. We finally analyze the complexity of our proposed algorithm.

3.1 Problem Formulation

The goal of our work is to identify the root causes of performance anomalies, including but not limited to 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.

3.2 Overview

The overview of KPIRoot+ is shown in Fig. 4, which consists of three key components, namely, time series decomposition-based 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 a newly designed Improved SAX representation to downsample the raw KPI. Then, KPIRoot detects the potential anomaly segments, including different anomaly types in the downsampled alarm KPI of the host cluster (Section 3.3). In this step, an anomaly score that describes the probability of the KPI being anomalous will be computed, and an anomaly segment will be automatically extracted around the spike. Then, KPIRoot+ conducts a similarity analysis to compute the similarity between VM KPIs and the alarm KPI during the anomaly period (Section 3.4). 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 3.5) 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.

Before detailing each component, we first clarify the two primary improvements of KPIRoot+ over the original KPIRoot framework, which directly address the limitations identified during its industrial deployment. First, we significantly enhance the anomaly detection capability. The original KPIRoot relied on a threshold-based method, effective for trend anomalies but insufficient for capturing the full spectrum of performance issues. KPIRoot+ introduces a more sophisticated approach using time series decomposition (STL) combined with a U-Net autoencoder. This allows for the distinct identification of trend, seasonal, and residual anomalies, providing a more comprehensive and accurate input for the subsequent root cause analysis, as will be detailed in Section 3.3. Second, we refine the similarity analy-

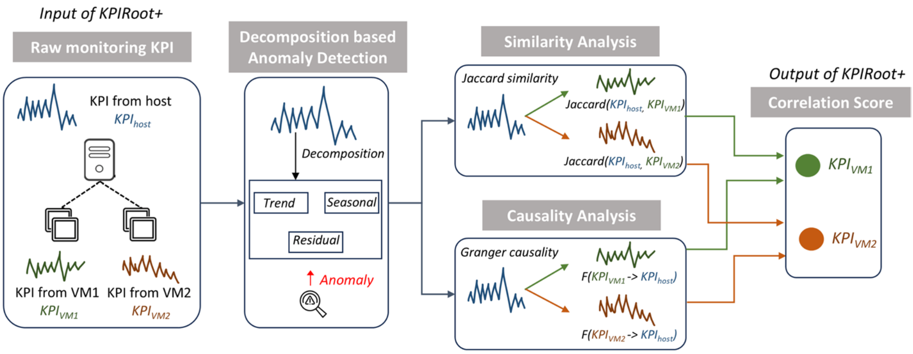


Fig. 4 The Overview of Our Proposed Method KPIRoot+

sis by replacing the standard Symbolic Aggregate Approximation (SAX) with an Improved SAX (ISAX). The original SAX representation could obscure critical details by only considering a segment's average value. ISAX enriches this representation by incorporating trend information, ensuring that KPI segments with different behaviors (e.g., one rising, one falling) are not incorrectly mapped to the same symbol. This leads to a more precise and discriminative similarity measurement, as elaborated in Section 3.4.

3.3 Time Series Decomposition Based Anomaly Segment Detection

To be efficient and meet the industrial requirement of real-time identification, KPIRoot (Gu et al. 2024b) proposes to adopt Symbolic Aggregate Approximation (SAX) (Lin et al. 2003). 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 (Minnen et al. 2007). 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" (Senin and Malinchik 2013). 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 (Guo et al. 2010). 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 (Lin et al. 2003). Before converting it to the PAA, we normalize each KPI to have a mean of zero and a standard deviation of one. However, SAX representation can obscure significant variation trends due to its reliance on segment averages, potentially leading to inaccurate representations.

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, which can cause performance issues for the system. However, in a dynamic cloud system, the threshold at which 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 (Zhao et al. 2020a). KPIRoot assumes that by detecting an uprush in workload, the early warning of a 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 $x_{t_e-1} > x_{t_s}$.

However, KPIRoot (Gu et al. 2024b) primarily targets trend anomalies through threshold-based techniques, which may fall short in identifying performance anomalies in large-scale cloud systems. The complexity and scale can lead to multiple overlapping types of performance anomalies, including level shifts, periodic variations, and sudden spikes or dips.

Time Series Decomposition Based Anomaly Detection We extend our previous work by proposing to utilize time series decomposition to better differentiate and detect these diverse anomaly types. This method distinctly identifies and addresses performance anomalies, which can often be obscured in a unified analysis. We assume the metric time series can be decomposed as the sum of three different components, namely, trend, seasonality, and remainder components:

$$X_{host}^t = \tau_{host}^t + s_{host}^t + r_{host}^t, t = 1, 2, ..., n \tag{3}$$

where X_{host}^t denotes the original host cluster KPI at time t, τ_{host}^t denotes the trend, s_{host}^t denotes the periodic component and r_{host}^t is the residual component.

In this paper, we propose to use the Seasonal and Trend decomposition using the Loess (STL) algorithm, which is a robust and versatile method for decomposing time series data (RB 1990). It uses a sequence of Loess (locally estimated scatter plot smoothing) regressions. The flexibility of STL in handling various seasonal patterns and the ability to adjust its parameters make it particularly suitable for complex and non-linear metrics in large-scale cloud systems.

After obtaining the decomposition into seasonal, trend, and remainder components, we perform anomaly detection on each component separately to identify distinct types of anomalies. To encode the complex patterns of the time series, it is necessary to consider both the local and global information, *i.e.*, multi-scale features. We adopt an auto-encoder network architecture with skip connections, also known as the U-Net structure (Ronneberger et al. 2015). It is trained on multiple sliding window segments of the monitoring metrics. Although the autoencoder approach may incur some additional computational cost, it remains affordable, considering there is typically only one alarm KPI against thousands of VM KPIs.

3.4 Similarity Analysis

Motivated by Yang et al. (2021), 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 (Yang et al. 2021), HALO (Zhang et al. 2021), and CMMD (Yan et al. 2022), 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.

KPIRoot transforms the KPIs into symbolic sequences and then computes the similarity between these sequences using the Jaccard similarity coefficient. A discretization technique that produces symbols with equal probability is used to obtain the discrete representation with symbols. As proved by Lin et al. (2003), the normalized KPIs have nearly Gaussian distributions. It is easy to pick equal-sized areas under the Gaussian distribution curve using lookup tables for the cut line coordinates, slicing the area under the Gaussian curve. Suppose there 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 the Section 3.3 between β_i to β_{i+1} will be assigned with the i^{th} symbol shown as follows:

$$s_i = alphabet_l, \quad if\ \beta_l \leq p_i \leq \beta_{l+1} \tag{4}$$

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. 5.

However, the traditional Symbolic Aggregate Approximation (SAX) method, while effective for dimensionality reduction, has a critical limitation in the context of root cause analysis: its reliance on segment averages can obscure significant underlying trends. This can lead to ambiguity where KPI segments with distinctly different behaviors are mapped to the same symbolic representation, potentially misleading the analysis. For instance, consider a scenario involving two VMs: one is recovering from a heavy load (its CPU usage is

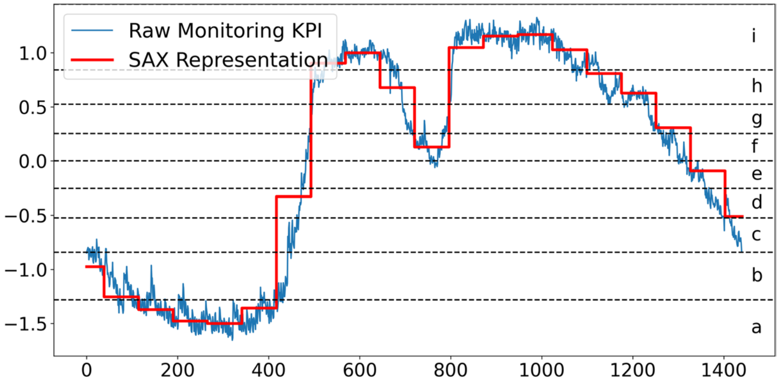


Fig. 5 An Illustration of SAX Representation

decreasing), while the other is beginning to malfunction, causing its CPU usage to escalate. Although their diagnostic implications are opposite, their average CPU usage over a short window could be identical. Standard SAX would incorrectly assign them the same symbol, masking the emerging problem.

Improved SAX representation (ISAX) To address this, an Improved SAX representation (ISAX) is proposed (shown in Fig. 6), which incorporates the variation trend indicators. This exact limitation and the benefit of our improvement are illustrated in Fig. 7. As shown, two KPI segments with opposite trends are processed. It demonstrates how standard SAX, by considering only the mean value, incorrectly maps both the "recovering" and "degrading" segments to the same ambiguous symbol. To address this deficiency, KPIRoot+ introduces an Improved SAX representation. By incorporating trend information, ISAX correctly assigns distinct and more descriptive symbols ‘$(c, \downarrow)$‘ and ‘$(c, \uparrow)$‘ to each segment. This preserves the critical diagnostic information needed for accurate root cause localization.

To maintain the efficiency of the approach, only the primary trend direction (upward, downward, or stable) is considered, ensuring that the enhanced representation remains computationally feasible while providing more critical insights into the KPI’s dynamic behavior.

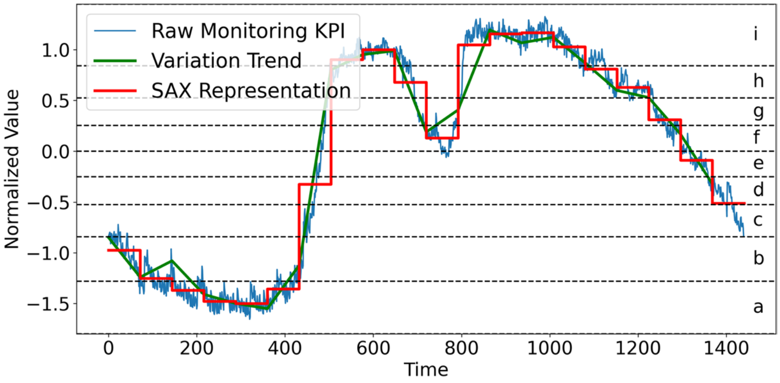


Fig. 6 An Illustration of Improved SAX Representation

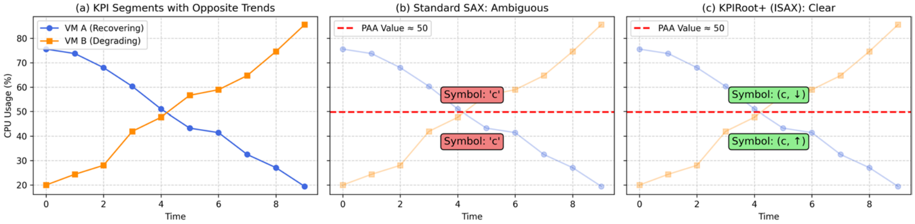


Fig. 7 Illustration of the standard SAX limitation and the ISAX improvement

The trend information, represented by the sign of the slope within the dimensionality reduction window, is calculated as follows:

$$\phi_i = sgn(x_{\frac{n}{w}\cdot i} - x_{\frac{n}{w}\cdot(i-1)+1}) \tag{5}$$

where $x_{\frac{n}{w}\cdot i}$ and $x_{\frac{n}{w}\cdot(i-1)+1}$ are the start and end point of the i^{th} metric segment in PAA representation. The Improved SAX in KPIRoot+ will further differentiate between two metric segments that originally map to the same symbol under traditional SAX due to similar averages. By incorporating the variation trend, Improved SAX assigns different symbols to segments that have the same average but different trends, such as an increasing versus a decreasing sequence. Different from the original SAX representation, improved SAX will assign the PAA representation element between β_i to β_{i+1} a symbol as follows:

$$s_i = alphabet_{2\alpha-\phi_i\cdot l}, \quad if\ \beta_l \leq p_i \leq \beta_{l+1} \tag{6}$$

We adopt the Jaccard similarity coefficient rather than other similarity measures because of its advantages when dealing with symbolic sequences like the SAX representation (He et al. 2016). 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:

$$Jaccard(S_{host}, S_i) = \frac{|S_{host} \cap S_i|}{|S_{host} \cup S_i|} \tag{7}$$

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.

The Improved Symbolic Aggregate Approximation method is effective in reducing the dimension of raw KPI while preventing trend information loss; however, the computation of the Improved SAX representation-based similarity does not provide any insights into the causality between VM KPIs and alarm KPIs. As mentioned by Mariani et al. (2020), 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 (Shojaie and Fox 2022) as a supplement, which will be elaborated in the following section.

3.5 Causality Analysis

Granger Causality is a statistical hypothesis test used to determine if one KPI is useful in forecasting another KPI (Arnold et al. 2007). 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:

$$p^t_{alarm} = \hat{a_0} + \sum_{j=1}^{q} \hat{a_j} p^{t-j}_{alarm} + \hat{\varepsilon_t} \tag{8}$$

$$p^t_{alarm} = a_0 + \sum_{j=1}^{q} a_j p^{t-j}_{alarm} + \sum_{j=1}^{q} b_j p^{t-j}_i + \varepsilon_t \tag{9}$$

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:

$$F_{vm \to host} = \frac{\sum_{t=t_s+q}^{t_e} (\hat{\varepsilon}^2_t - \varepsilon^2_t)/q}{\sum_{t=t_s+q}^{t_e} \varepsilon^2_t/(t_e - t_s - 2q - 1)} \tag{10}$$

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:

$$c_i = \lambda \times Jaccard(S_{host}, S_i) + (1 - \lambda) \times F_{vm \to host} \tag{11}$$

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

3.6 Complexity Analysis and Comparison

The proposed method KPIRoot+ is summarized in Algorithm 1. To address the efficiency requirements of industrial applications, both KPIRoot and our proposed KPIRoot+ are designed to be highly efficient. The core analysis loop of the original KPIRoot involves similarity and causality analysis on downsampled KPIs. As described in our preliminary work, we set the length of the SAX representation $w \approx \sqrt{n}$. The time complexity of generating the SAX representation and computing Jaccard similarity is $\mathcal{O}(\sqrt{n})$. The complexity of Granger causality is dominated by the autoregression model, which is $\mathcal{O}(\sqrt{n} \times q^3)$, where q is the time lag. Therefore, the complexity for analyzing each VM KPI in KPIRoot is $\mathcal{O}(\sqrt{n} \times (q^3 + 2))$. This efficiency is a key advantage over methods like AID (based on DTW), which has a complexity of $\mathcal{O}(n^2)$. KPIRoot+ enhances this framework by introducing a more advanced anomaly detection module and the Improved SAX (ISAX) representation. The time series decomposition and U-Net autoencoder for anomaly detection are applied *only once* to the single alarm KPI, incurring a fixed, one-time computational cost of that does not scale with the number of VMs. Similarly, while ISAX incorporates trend information, the complexity of its generation and the subsequent Jaccard similarity calculation remains $\mathcal{O}(\sqrt{n})$. Consequently, the per-VM analysis loop in KPIRoot+ retains the same asymptotic complexity of $\mathcal{O}(\sqrt{n} \times (q^3 + 2))$ as the original KPIRoot. This demonstrates that KPIRoot+ achieves higher accuracy with only a negligible impact on its overall scalability, making it equally suitable for industrial applications that demand real-time root cause localization.

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 $c_i$
for $i = 1; i \le w; i++$ do
    $p^i_{alarm} = \frac{w}{n}\sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} x^j_{alarm}$
    $\phi^i_{alarm} = sgn(x^{\frac{n}{w}\cdot i}_{alarm} - x^{\frac{n}{w}\cdot(i-1)+1}_{alarm})$
end for
// Anomaly Segment Detection
$X^t_{alarm} = \tau^t_{host} + s^t_{host} + r^t_{host}$
$i_{anomaly} = AE(\tau_{host}) \cup AE(s_{host}) \cup AE(r_{host})$
$p_{alarm} = p_{alarm}[i_{anomaly}]$
$s^i_{alarm} = \{alphabet_{2\alpha-\phi_i \cdot l},\quad if\ \beta_l \le p^i_{alarm} \le \beta_{l+1}\}$
for $i = 1; i \le m; i++$ do
    // Similarity Analysis
    for $k = 1; k < m; k++$ do
        $p^k_i = \frac{w}{n}\sum_{j=\frac{n}{w}(k-1)+1}^{\frac{n}{w}k} x^k_i$
        $p_i = p_i[i_{anomaly}]$
        $\phi^k_i = sgn(x_i^{\frac{n}{w}\cdot k} - x_i^{\frac{n}{w}\cdot(k-1)+1})$
        $s^k_i = \{alphabet_{2\alpha-\phi_i \cdot l},\ s.t.\ \beta_l \le p^k_i \le \beta_{l+1}\}$
    end for
    $Jaccard(S_{host}, S_i) = \frac{|S_{host} \cap S_i|}{|S_{host} \cup S_i|}$
    // Causality Analysis
    for $t = t_s + q; t < t_e; t++$ do
        $p^t_{alarm} = \hat{a}_0 + \sum_{j=1}^{q} \hat{a}_j p^{t-j}_{alarm} + \hat{\varepsilon}_t$
        $p^t_{alarm} = a_0 + \sum_{j=1}^{q} a_j p^{t-j}_{alarm} + \sum_{j=1}^{q} b_j p^{t-j}_i + \varepsilon_t$
    end for
    $F_{vm \to host} = \frac{\sum_{t=t_s+q}^{te}(\hat{\varepsilon}_t^2 - \varepsilon_t^2)/q}{\sum_{t=t_s+q}^{te}\varepsilon_t^2/(t_e - t_s - 2q - 1)}$
    $c_i = \lambda \times Jaccard(S_{host}, S_i) + (1 - \lambda) \times F_{vm \to host}$
end for
return $c_i$
```

4 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+ in performance issue detection compared with baselines?
- RQ2: How effective is KPIRoot+ compared with KPI root cause localization baselines?
- RQ3: How effective is each component of KPIRoot+ in root cause localization?
- RQ4: How efficient is KPIRoot+ in localizing root cause KPIs compared to baselines?
- RQ5: How sensitive is KPIRoot+ to each hyperparameter?

4.1 Experiment Setting

4.1.1 Datasets

To confirm the practical significance of KPIRoot+ , we conduct our evaluation on the same three large-scale industrial datasets from Cloud $\mathcal{H}$ that were used in the evaluation of the original KPIRoot (Gu et al. 2024b). Three datasets from large-scale online services in three Available Zones (AZs) of Cloud $\mathcal{H}$ are collected. The statistics of three industrial datasets are shown in Table 1. 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. This allows for a direct and rigorous comparison of the improvements offered by our new approach KPIRoot+ .

4.1.2 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 the top *k* recommended results by root cause localization methods. Hit Rate@*k* is a widely used metric to measure whether the

Table 1 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

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.

Additionally, we propose to use the Normalized Discounted Cumulative Gain (NDCG) in our evaluation metrics, specifically NDCG@10. NDCG is more beneficial because it considers the rank position of each result, applying a discounting factor to lower-ranked positions, which measures how easily engineers can find the culprit VMs. This is crucial in our scenarios as the most relevant root causes are more prioritized for investigation. NDCG@1 is left out because it is the same as Hit Rate@1 in our scenario. NDCG@*k* measures the extent to which the root cause appears higher up in the ranked candidate list. Thus, the higher the above measurements, the better.

4.2 Experimental Results

4.2.1 RQ1 The effectiveness of KPIRoot+ in performance issue detection

To answer this research question, we compare the performance of KPIRoot+ with a comprehensive set of baselines, including classical methods 3σ (Yu et al. 2023), LOF (Local Outlier Factor) (Breunig et al. 2000), IF (Isolation Forest) (Liu et al. 2008); clustering methods, BIRCH (Zhang et al. 1996); statistical approaches, SPOT (Siffer et al. 2017), BARO (Pham et al. 2024); a suite of deep learning models, Autoencoder (Li et al. 2023), LSTM (Lin et al. 2018), Donut (Xu et al. 2018), DAGMM (Zong et al. 2018), OmniAnomaly (Su et al. 2019a), SRCNN (Ren et al. 2019), GDN (Deng and Hooi 2021), TranAD (Tuli et al. 2022); as well as our previous work, KPIRoot (Gu et al. 2024b). The results are shown in Table 2, where the best scores are in bold. As is evident, KPIRoot+ achieves the highest Precision, Recall, and F1 scores across all three datasets, demonstrating its superior performance.

Table 2 Experimental Results of Different Anomaly Detection Methods

Methods	Dataset A			Dataset B			Dataset C		
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
3σ	0.709	0.762	0.738	0.765	0.694	0.730	0.797	0.685	0.747
LOF	0.681	0.587	0.753	0.619	0.591	0.737	0.681	0.598	0.715
IF	0.699	0.612	0.788	0.673	0.607	0.772	0.715	0.612	0.706
Autoencoder	0.791	0.770	0.782	0.776	0.810	0.794	0.859	0.793	0.823
Donut	0.813	0.829	0.823	0.795	0.824	0.806	0.798	0.875	0.854
DAGMM	0.869	0.847	0.851	0.816	0.798	0.807	0.829	0.845	0.841
OmniAnomaly	0.867	0.844	0.852	0.866	0.879	0.871	0.809	0.833	0.821
SRCNN	0.855	0.807	0.834	0.843	0.793	0.820	0.841	0.836	0.840
GDN	0.812	0.787	0.802	0.819	0.769	0.797	0.843	0.827	0.830
TranAD	0.851	0.878	0.864	0.872	0.849	0.857	0.849	0.796	0.833
LSTM	0.836	0.752	0.805	0.826	0.865	0.824	0.829	0.863	0.839
BIRCH	0.713	0.682	0.707	0.684	0.719	0.697	0.736	0.703	0.715
SPOT	0.746	0.787	0.751	0.720	0.776	0.735	0.785	0.743	0.761
BARO	0.794	0.724	0.746	0.812	0.741	0.794	0.717	0.767	0.738
KPIRoot	0.787	0.712	0.755	0.782	0.717	0.744	0.803	0.709	0.759
KPIRoot+	**0.914**	**0.943**	**0.928**	**0.924**	**0.907**	**0.913**	**0.942**	**0.863**	**0.894**

Our analysis of the baselines reveals distinct performance tiers. The classical methods like 3σ, LOF, and IF are effective at detecting point-wise residual anomalies but are less suited for persistent ones. Similarly, the clustering-based BIRCH and statistical SPOT and BARO provide a foundational but limited performance. The deep learning models, however, show an improvement over these methods. Autoencoder and LSTM models are adept at capturing deviations from historical patterns, making them also effective for seasonal anomalies. KPIRoot is specifically designed for trend anomalies by comparing observation windows, thus fall suboptimal.

Among the more advanced deep learning baselines, a clear performance hierarchy emerges. The Transformer-based TranAD and the RNN-based OmniAnomaly consistently deliver the strongest results among all baselines, showcasing the power of models that explicitly capture complex temporal dependencies. Following them, reconstruction-based models like the VAE-based Donut and the Autoencoder-based DAGMM, along with the SRCNN, also provide robust performance by effectively learning the normal patterns. The GNN-based GDN, while still outperforming classical methods, shows slightly lower performance in this context, which is expected as its core strength–modeling inter-metric relationships–cannot be fully utilized when conducting anomaly detection on a single KPI, as is the case in our experiments. Despite the strong performance of top-tier models like TranAD and OmniAnomaly, they still fall short of KPIRoot+ . These models learn a single, holistic representation of the time series, which makes it difficult to differentiate and adapt to the distinct characteristics of trend, seasonal, and residual anomalies when they are mixed together. This is where KPIRoot+ demonstrates its superiority. By utilizing a time series decomposition-based method, KPIRoot+ first isolates these components and then analyzes them with tailored strategies. This leads to a more nuanced and accurate identification of all anomaly types, resulting in higher overall accuracy and providing a more reliable foundation for subsequent root cause localization.

4.2.2 RQ2 The effectiveness of KPIRoot+ in root cause localization

To answer this research question, we compare the performance of KPIRoot+ with a comprehensive set of baselines. These include correlation-based methods (Kendall, Spearman, CloudScout (Yin et al. 2016)); advanced statistical and causal approaches like the statistical test-based ϵ-Diagnosis (Shan et al. 2019), the change point detection-based BARO (Pham et al. 2024), and the causal inference methods CIRCA (Li et al. 2022) and RCD (Ikram et al. 2022); and other established techniques such as AID (Yang et al. 2021), which uses DTW distance, LOUD (Mariani et al. 2018), a graph centrality-based method, HALO (Zhang et al. 2021), which employs conditional entropy, the GNN-based CMMD (Yan et al. 2022), and our previous work, KPIRoot (Gu et al. 2024b). Table 3 presents the results, highlighting the best F1, Hit@5, Hit@10, and N@10 scores in bold. We observe that KPIRoot+ consistently outperforms all baseline methods across all metrics on the three datasets. In particular, the improvement achieved by KPIRoot+ is more pronounced in Dataset B and Dataset C. This is because these datasets focus on KPIs, such as request rates, related to the load balancer, which manages the distribution of network traffic across physical machines. As a result, anomalies in VM request rates tend to precede anomalies in host clusters, providing an early indicator for potential issues. It is important to note that, as shown in Table 3, the number of root causes often exceeds 5. Despite this, achieving Hit@5 scores exceeding

Table 3 Experimental Results of Different Root Cause Localization Methods

Methods	Dataset A				Dataset B				Dataset C			
	F1	H@5	H@10	N@10	F1	H@5	H@10	N@10	F1	H@5	H@10	N@10
Kendall	0.651	0.562	0.728	0.507	0.605	0.594	0.770	0.546	0.657	0.635	0.727	0.651
Spearman	0.681	0.587	0.753	0.518	0.619	0.591	0.737	0.577	0.681	0.598	0.715	0.636
CloudScout	0.699	0.612	0.788	0.657	0.673	0.607	0.772	0.683	0.715	0.612	0.706	0.608
LOUD	0.736	0.652	0.813	0.624	0.736	0.625	0.824	0.657	0.709	0.653	0.829	0.689
AID	0.746	0.652	0.749	0.634	0.673	0.618	0.794	0.602	0.665	0.613	0.729	0.597
ϵ-Diagnosis	0.707	0.594	0.766	0.668	0.644	0.622	0.749	0.611	0.697	0.628	0.810	0.643
BARO	0.797	0.753	0.827	0.708	0.715	0.695	0.864	0.681	0.709	0.696	0.826	0.679
CIRCA	0.706	0.679	0.760	0.695	0.675	0.718	0.823	0.734	0.749	0.670	0.801	0.674
RCD	0.749	0.628	0.787	0.712	0.691	0.628	0.757	0.703	0.692	0.688	0.837	0.667
HALO	0.734	0.651	0.842	0.667	0.632	0.569	0.811	0.598	0.719	0.635	0.789	0.646
CMMD	0.776	0.632	0.833	0.604	0.679	0.594	0.848	0.613	0.721	0.667	0.801	0.658
KPIRoot	0.859	0.731	0.909	0.766	0.860	0.749	0.946	0.779	0.829	0.713	0.895	0.741
KPIRoot+	**0.884**	**0.780**	**0.934**	**0.823**	**0.891**	**0.799**	**0.967**	**0.842**	**0.871**	**0.755**	**0.936**	**0.797**

75% is significant, as it indicates that our method accurately identifies a substantial portion of root causes within just the top 5 predictions. Additionally, the high F1 score and Hit@10 demonstrate the method's effectiveness for industrial applications.

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, as it can occur due to a shared underlying cause rather than a direct influence. The more advanced statistical and causal inference methods show improved, yet varied, performance. Statistical approaches like ϵ-Diagnosis and BARO rank metrics based on distribution changes during a failure. However, their effectiveness can be limited when many VMs exhibit simultaneous changes, making it difficult to isolate the true root cause from the symptoms. Causal inference methods offer a more direct approach. CIRCA, for example, uses a causal graph to identify the metric whose conditional probability changes most significantly. However, its potential is not fully realized here, as it requires a predefined architecture graph which is not applicable to our independent VMs. RCD is slightly more effective as it discovers causal relationships directly from data without needing a predefined graph. Though CMMD has the ability to capture complex, nonlinear relationships 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 somewhat alleviates the defect of neglecting causality. The LOUD method applies graph centrality, but its performance is highly dependent on the graph construction, making it less effective in our context. KPIRoot incorporates both similarity analysis (SAX representation) and causality analysis (Granger causality test), leading to better root cause localization accuracy than other baselines. Compared with KPIRoot, KPIRoot+ can identify performance anomalies more accurately and comprehensively, thereby enhancing the accuracy of subsequent root cause analysis. The improved SAX technique utilized by KPIRoot+ helps retain trend variation information, reducing false positives and enhancing the overall robustness of the entire pipeline.

4.2.3 RQ3 The effectiveness of components in KPIRoot+

To answer this research question, we conducted an ablation study on KPIRoot+ . We compared two baseline models, removing the Improved SAX and Decomposition-based anomaly detection part of KPIRoot+ to investigate the contribution of these two designs.

- *KPIRoot+ w/o I* This baseline removes the Improved SAX and utilizes the SAX representation in KPIRoot. The Decomposition-based anomaly detection is adopted.
- *KPIRoot+ w/o D* This baseline removes the Decomposition-based anomaly detection and utilizes the Improved SAX representation to downsample the original metrics.

Table 4 shows the performance comparison between KPIRoot+ and its variants. In summary, the effectiveness of KPIRoot+ is enhanced with the utilization of Improved SAX and Decomposition-based anomaly detection. Indeed, the variant without the Improved SAX performs better than the variant without Decomposition-based anomaly detection. This is because accurate anomaly detection is crucial for conducting subsequent similarity and causality analyses, which are essential for correlating the true root cause. While the trend

Table 4 Experimental Results of the Ablation Study of KPIRoot+

Methods	Dataset A				Dataset B				Dataset C			
	F1	H@5	H@10	N@10	F1	H@5	H@10	N@10	F1	H@5	H@10	N@10
KPIRoot+ *w/o* I	0.872	0.763	0.935	0.789	0.883	0.766	0.963	0.793	0.856	0.740	0.922	0.784
KPIRoot+ *w/o* D	0.865	0.752	0.926	0.770	0.872	0.762	0.958	0.784	0.845	0.734	0.909	0.766
KPIRoot+	**0.884**	**0.780**	**0.934**	**0.823**	**0.891**	**0.799**	**0.967**	**0.842**	**0.871**	**0.755**	**0.936**	**0.797**

information captured during downsampling by Improved SAX is also important, as the increasing or decreasing trend within a downsample window can sometimes be crucial for determining the true root cause, there are subtle differences between the variation trends of the true root cause and false positives. Both variants outperform the original KPIRoot, demonstrating that the integration of these two designs significantly boosts the root cause localization performance.

4.2.4 RQ4 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. 8, from which we can observe that KPIRoot is still the most efficient, with an average execution time of around only 5 seconds. While KPIRoot+ takes around 8 seconds, it is still capable of providing real-time analysis and delivers more accurate results, making it a worthwhile tradeoff. The additional overhead is primarily due to the decomposition-based anomaly detection method and the Improved SAX, which uses more symbols, making the similarity analysis more time-consuming. However, this overhead is absolutely acceptable, given the improved accuracy and comprehensiveness of the results. This indicates 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. 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 superior accuracy while not adding much more computational overhead, thereby offering an excellent balance between efficiency and precision in real-time root cause analysis. It is a highly promising tool for conducting real-time root cause analysis within large-scale cloud systems.

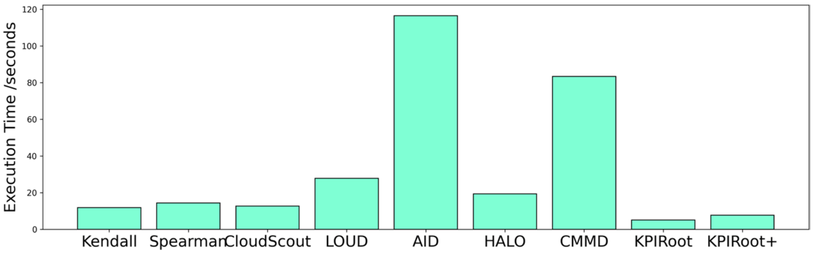
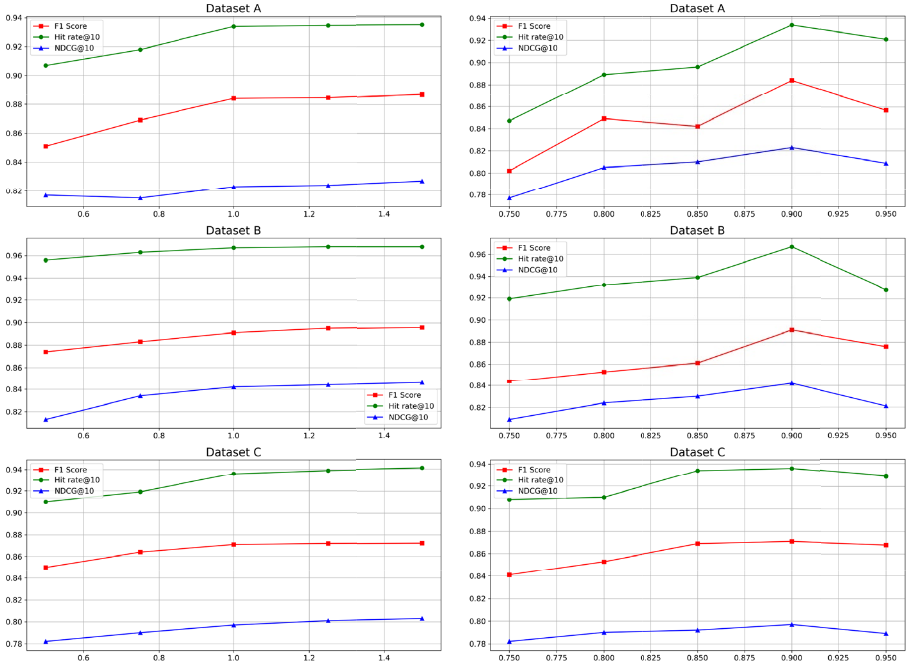


Fig. 8 Root Cause Localization Time for All Methods

4.2.5 RQ5 Sensitivity Analysis of KPIRoot+

The parameter w determines the dimension of the representation vector for Improved SAX, while λ is crucial for balancing the tradeoff between similarity and causality analysis. We evaluate the sensitivity of KPIRoot+ to these two hyperparameters using three industrial datasets. To ensure fairness, we vary the values of w and λ while keeping all other parameters constant. Specifically, w is chosen as a multiple of $\sqrt{n}$, ranging from 0.5 to 1.5 times $\sqrt{n}$. This selection ensures that we maintain an $O(\sqrt{n})$ time complexity, aligning with our efficiency goals. For λ, we select values from 0.75 to 0.95, acknowledging that the scale of similarity is typically smaller than that of causality. This choice effectively balances the tradeoff between similarity and causality within our analysis framework. By systematically adjusting these parameters, we aim to optimize the performance and robustness of our model across different datasets.

Figure 9 presents the experimental results of RQ5. For the parameter w, the performance is relatively stable between 1 and 1.5 times $\sqrt{n}$. If the dimension of Improved SAX is too low, there is more information loss during the downsampling process, which decreases accuracy. However, a larger dimension may cause the time complexity to increase quickly, and it may not significantly enhance performance beyond $\sqrt{n}$. Thus, it is reasonable to select w in this range to balance the tradeoff between computational efficiency and model accuracy. For the parameter λ, a good tradeoff between these two parts indeed helps improve the performance of KPIRoot+ . In Dataset C, this variation of performance due to the parameter is not so significant because either the similarity score or the causality score of the root cause is high. Thus, the tradeoff coefficient has a lower influence on the overall performance.



(a) Parameter Sensitivity of w

(b) Parameter Sensitivity of λ

Fig. 9 Parameter Sensitivity of KPIRoot+

5 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 was feasible given the limited scale of each cluster. Therefore, 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 and one hour to identify and mitigate the root causes.

To alleviate these issues, we have deployed KPIRoot+ in Cloud $\mathcal{H}$ since *Aug 2023*. 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. This overall deployment pipeline of deploying KPIRoot+ in Cloud $\mathcal{H}$ is depicted in Fig. 10. The software reliability engineers collect the monitoring metrics (like CPU usage, network traffic, memory usage) of the host clusters and each VM through monitoring tools like Grafana, Prometheus, etc (Agarwal et al. 2023). Then, these monitoring metrics are stored in the Data Lake of Huawei Cloud, a highly scalable and flexible storage system that consists of the Data Lake Storage, the Data Warehouse, and the Data Lake Governance

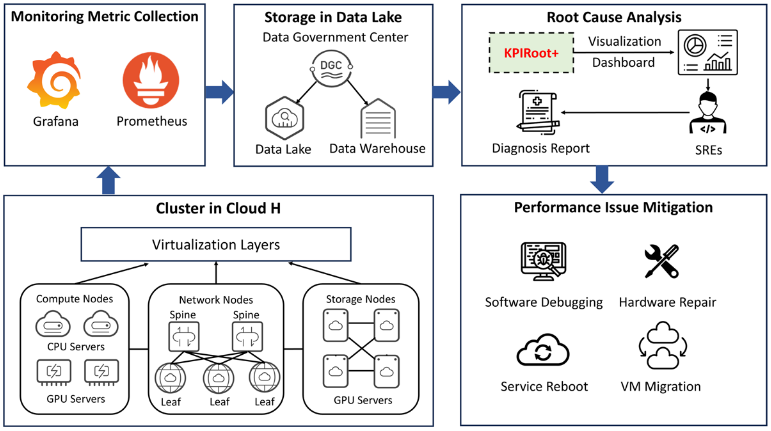


Fig. 10 The overall pipeline of deploying KPIRoot+ in Cloud $\mathcal{H}$

Center (DGC). Data Lake Storage is the actual storage space where all the data, including the monitoring metrics, is stored, while the Data Warehouse is an enterprise system used for reporting and data analysis. On top of these two components, the DGC is responsible for managing the data stored in the data lake and overseeing the lifecycle of the data, from ingestion and storage to usage and deletion. With Cloud $\mathcal{H}$'s data lake, real-time data analysis is enabled, *i.e.*, as soon as monitoring metrics are collected and stored in the data lake, they can be immediately accessed and analyzed by the performance issue diagnosis system empowered with KPIRoot+ . The results of KPIRoot+ are visualized on the dashboard and can be easily observed and understood by engineers. Once the root cause has been investigated and identified, the SREs prepare a diagnosis report, including a detailed description of the identified root cause and potential mitigation strategies.

In Fig. 11, the practical application of our previous root cause analysis tool, KPIRoot, in real industrial scenarios is shown. 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 KPI indicating the root cause 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 VM can be quickly localized by analyzing its KPI, generally within five minutes. With this result, we throttle the throughput of VM1 immediately after the alarm KPI is fired. As shown in Fig. 11, the overall traffic is limited, and the alarm KPI returns to a normal range quickly. However, KPIRoot primarily considers trend anomalies and sometimes neglects critical performance anomaly information, leading it to recommend inaccurate root causes. A case that we identified is shown in Fig. 12. The frequent packet loss suggested that the load bal-

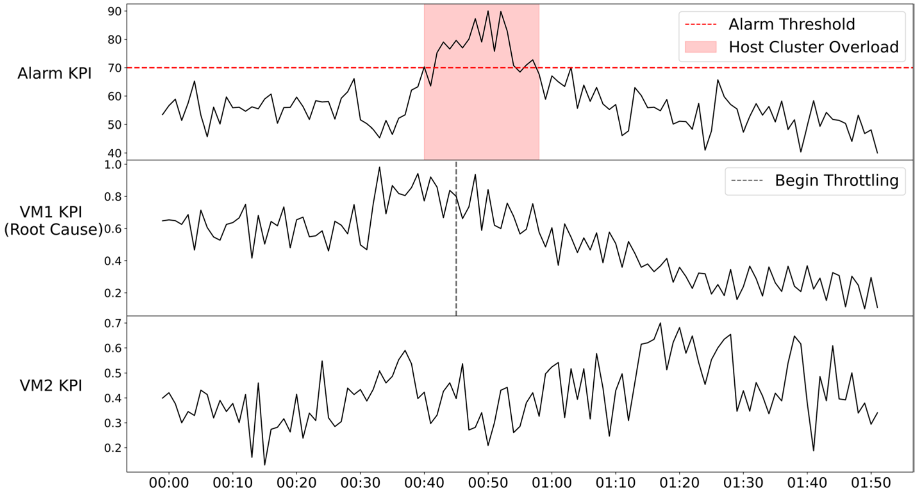


Fig. 11 Case Study of KPIRoot

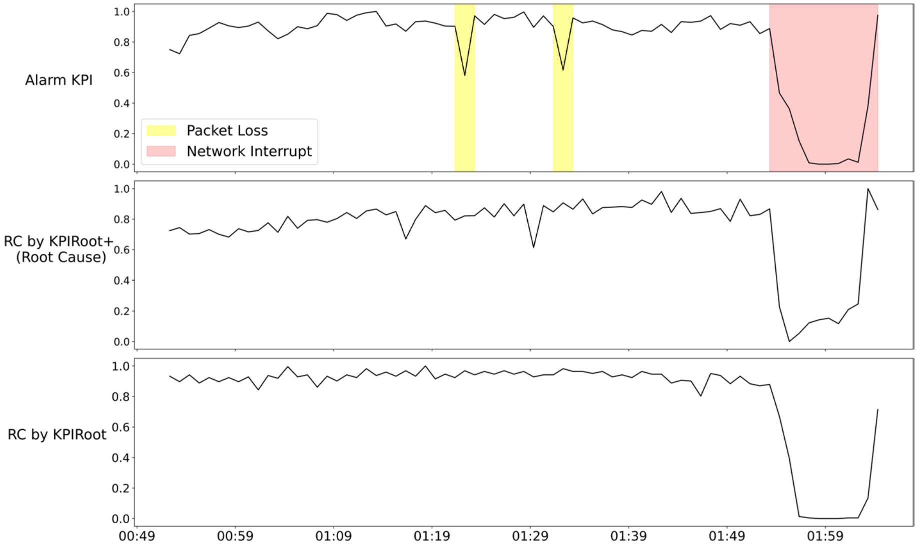


Fig. 12 Case Study of KPIRoot+ 1

ancer was not distributing traffic evenly, causing the VM corresponding to the second KPI to experience frequent network congestion. As requests accumulated, it led to more severe network interruptions, which are symptomatic of overwhelming network buffers and potential misconfigurations in the load-balancing algorithm. This buildup of unsent packets can cause buffer overflow and increased latency, resulting in temporary network interruptions. Therefore, the VM corresponding to the second KPI was indeed the root cause, while the drop in the third KPI was a passive consequence and its associated VM was not the root cause. These two packet loss anomalies, indicated by transient KPI drops in the alarm KPI, would be ignored by KPIRoot. KPIRoot will instead recommend the VM associated with the third KPI as the root cause. In contrast, KPIRoot+ considers these types of anomalies, correctly identifying the VM linked to the second KPI as the root cause.

However, while KPIRoot+ generally improves accuracy, its enhanced sensitivity is not without trade-offs. We also identified specific scenarios where this sensitivity could lead to less optimal results compared to the original KPIRoot. A representative example is illustrated in Fig. 13. In this instance, a significant trend anomaly in the alarm KPI indicates a true network interruption. The time series decomposition in KPIRoot+ , designed to capture various anomaly types, also identified minor residual anomalies within the data of another VM, as labeled in the yellow area. These residuals, in reality, represented acceptable operational jitter and were not indicative of a performance issue. However, KPIRoot+ ’s model assigned a notable score to these minor fluctuations, causing it to incorrectly flag the VM with jitter as a potential root cause. In contrast, the original KPIRoot, by exclusively focusing on significant trend shifts, correctly ignored the minor jitter. It successfully identified the VM whose KPI trend directly correlated with the major network interruption, thus providing the correct diagnosis in this specific case. This case highlights a key limitation: KPIRoot+ ’s sensitivity to residual anomalies can sometimes overshadow major trend-based events, a scenario where the simpler, trend-focused approach of the original KPIRoot proves more robust.

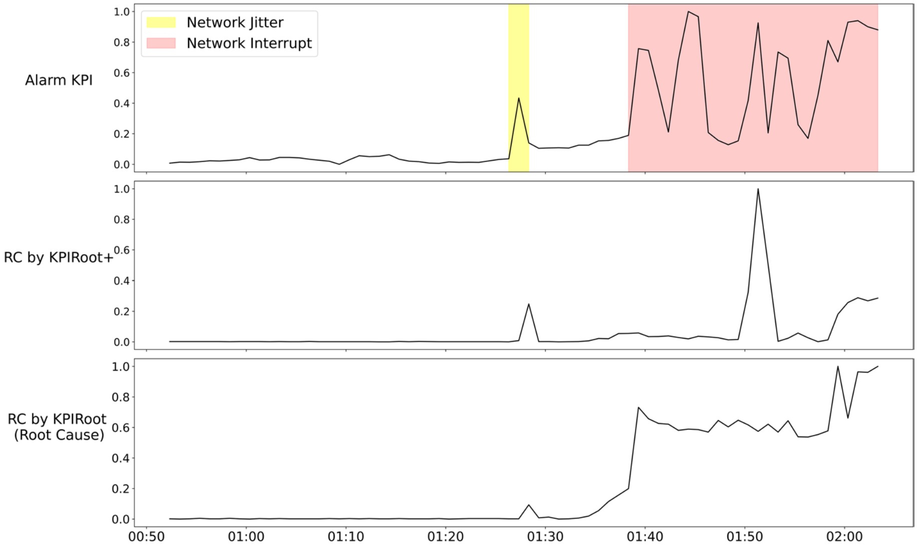


Fig. 13 Case Study of KPIRoot+ 2

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.

6 Discussion

In this section, we discuss the difference between our approach and existing root cause analysis approaches for microservice systems, the adaptability of our framework to dynamic environments, and why they are not applicable in our industrial scenario. We also identify some potential threats to the validity of our study.

6.1 Root Cause Analysis for Microservice System

Our objective shares some similarities with root cause analysis in microservice systems; however, several main differences exist in 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 (Chen et al. 2023) and ServiceRank (Ma et al. 2021) 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 (Wang et al. 2023b), 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.

6.2 Adaptability to Dynamic Environments

Adaptability is critical for any AIOps framework intended for deployment in large-scale, dynamic cloud systems where data patterns and system behaviors can evolve over time. We discuss the adaptability of our framework by examining its two core components. First, the time series decomposition-based anomaly segment detection module, which leverages time series decomposition and a U-Net autoencoder, is a learning-based component. Like any deep learning model, it is susceptible to concept drift if the statistical properties of the KPI data change significantly from the training data. To address this, we employ a periodic retraining strategy. The anomaly detection model is retrained weekly in our industrial deployment at Cloud $\mathcal{H}$. This process is highly efficient and lightweight. Since the alarm KPIs are collected at a one-minute granularity and the number of high-level alarm KPIs to monitor is typically in the low hundreds, the entire retraining process on Cloud $\mathcal{H}$'s infrastructure completes in less than one minute. This ensures the detection model remains up-to-date with the latest system behaviors without imposing a significant computational burden. Second, the root cause localization module performs similarity and causality analysis and is entirely deterministic and training-free. This component is inherently data-agnostic; it does not rely on learned historical patterns to function. Instead, it calculates similarity (via Improved SAX) and causality (via Granger causality) directly on the slice of anomalous data provided in real time. Consequently, this module does not suffer from concept drift and automatically adapts to new patterns or relationships in the system's KPIs.

Thus, the hybrid design of KPIRoot+ that combines a lightweight, periodically retrained detection model with a deterministic, training-free localization algorithm makes the framework robust and highly suitable for deployment in dynamic industrial environments where system conditions frequently change.

6.3 Robustness to Complex Multi-KPI Anomaly Scenarios

A consideration in real-world cloud environments is the scenario where a single fault triggers anomalies across numerous KPIs, and the root cause metric may not be the first to exhibit anomalous behavior. Our framework is designed with these practical challenges in mind.

First, our approach of initiating analysis from a single, highly-aggregated alarm KPI (e.g., total CPU utilization of a host cluster) is a deliberate design choice that reflects common operational practices. This strategy is particularly effective for identifying systemic issues like resource overload or network saturation, where a single, high-level metric serves as a reliable alarm trigger. While many individual VM KPIs will also show anomalies during such an incident, monitoring all of them for initial alerts is often impractical. It can be computationally expensive and lead to significant "alert fatigue" from noisy or transient fluctuations, making it difficult to distinguish a true incident from operational noise. Therefore, using a stable, golden, high-level KPI as the trigger provides a clear and efficient entry point for diagnosis.

Second, and more critically, our framework is explicitly designed to handle the challenging case where the root cause anomaly does not appear first chronologically. Relying purely on temporal precedence would fail in such cases. This is precisely why KPIRoot+ 's root cause ranking is based on a hybrid score that intelligently combines both causality and similarity. In a situation where a root cause VM's anomaly is slightly delayed, its time-series pattern will still exhibit a very high shape similarity to the aggregated host KPI's anomaly. Our scoring mechanism gives significant weight to this strong similarity, enabling it to compensate for the lack of strict temporal precedence. This allows KPIRoot+ to correctly identify and highly rank the true root cause, ensuring that our method remains robust and effective in these more complex and realistic failure scenarios. Furthermore, our method recommends a list of more than the top-5 potential root causes rather than a single one. This provides a practical safety net, making it highly feasible for an operator to capture the true root cause even if it is not ranked first in the list.

6.4 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 parameter settings.

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.

7 Related Work

7.1 Anomaly Detection in Cloud Systems

Ensuring the optimal performance of cloud systems is imperative. Monitoring KPIs are used to perceive the status of cloud systems and facilitate analysis when performance anomalies occur. Related studies can be categorized into statistic-based, machine learning-based, and deep learning-based approaches.

A variety of statistical and machine learning techniques have been proposed for anomaly detection. A common baseline is N-Sigma (or the 3-sigma rule), which assumes data follows a normal distribution and flags any point falling outside a three-standard-deviation range as an anomaly (Li et al. 2022). Clustering-based methods identify anomalies as points that do not belong to dense clusters. Examples include BIRCH (Zhang et al. 1996), an efficient online clustering algorithm that builds a compact tree-based summary of the data, making it suitable for large-scale systems and adopted in RCA tools like MicroDiag (Wu et al. 2021). Density-based methods like LOF (Local Outlier Factor) (Breunig et al. 2000) identify outliers by measuring the local density deviation of a data point with respect to its neighbors. Ensemble methods like Isolation Forest (iForest) (Liu et al. 2008) build multiple isolation trees under the assumption that anomalies are rare and easier to isolate. Other approaches leverage more advanced statistical theories. SPOT (Streaming Peaks-Over-Threshold) (Siffer et al. 2017) is a streaming anomaly detection method based on Extreme Value Theory (EVT). It models the tail of the data distribution to automatically set thresholds for rare events, making it robust to different underlying data distributions. Another recent approach, BARO (Pham et al. 2024), frames the problem as change point detection, employing multivariate Bayesian online change point detection to identify significant shifts in the joint distribution of metrics, thereby capturing dependencies between time series.

Recently, a variety of studies applying deep learning to conduct anomaly detection on metrics data have been proposed. Many of these are reconstruction-based, where a model learns to reconstruct normal data, and a high reconstruction error signals an anomaly. Foundational models in this category include the Autoencoder (AE) and Variational Autoencoder (VAE). For example, Donut (Xu et al. 2018) proposed an unsupervised VAE-based algorithm specifically for detecting anomalies in seasonal KPIs from large-scale internet services. To improve upon basic autoencoders, some works combine them with other statistical models. DAGMM (Zong et al. 2018) enhances the AE framework by jointly training it with a Gaussian Mixture Model (GMM) in an end-to-end fashion, allowing it to better model the density of the data in the latent space. While autoencoders capture feature patterns, they need enhancements to model temporal dependencies effectively. Thus, many approaches integrate Recurrent Neural Networks (RNNs) like LSTM and GRU. For instance, an LSTM model for temporal data combined with a Random Forest model for spatial data to predict node failures in cloud systems has been proposed (Lin et al. 2018). It uses a cost-sensitive function to handle imbalanced data. Similarly, OmniAnomaly (Su et al. 2019a) uses a stochastic RNN to model normal patterns and reconstruct time series, identifying anomalies based on reconstruction probabilities. Other hybrid approaches also exist, including SRCNN (Ren et al. 2019), an algorithm based on Spectral Residual (SR) combined with a Convolutional Neural Network (CNN) to create an efficient and accurate detector for production systems. Beyond temporal dependencies, capturing the complex relationships between metrics is crucial. To this end, some methods utilize temporal graph neural networks (GNNs) to address the relationships between metrics more explicitly. These approaches model the multivariate metrics as a graph where nodes are metrics and edges represent their dependencies. For instance, GDN (Graph Deviation Network) (Deng and Hooi 2021) proposes a GNN-based approach that learns the dependency graph from data using sensor embeddings and then uses a graph attention mechanism to forecast future values. Anomalies are identified as significant deviations from these forecasted values. To overcome the limitations of recurrent models, other works have adopted the Transformer architecture. TranAD (Tuli et al. 2022) leverages Transformer networks for faster, parallelizable training and better modeling of long sequences. It introduces an adversarial training procedure and a self-conditioning mechanism using "focus scores" to amplify small deviations and improve stability, making it robust even with limited training data.

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.

7.2 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, engineers can troubleshoot these specific parts of the system to alleviate the performance anomaly. RCA approaches can be broadly categorized into those based on correlation, causal inference, and attribute-level search.

A primary line of work identifies root causes by finding metrics that are most correlated with a failure or exhibit the most significant statistical change. For instance, CloudScout (Yin et al. 2016) employs the Pearson Correlation Coefficient on infrastructure-level KPIs (*e.g.*, CPU usage) to calculate the similarity between services. Similarly, AID (Yang et al. 2021) measures the "intensity of dependencies" by calculating the similarity between the status KPIs of caller and callee services. LOUD (Mariani et al. 2018) builds on this by applying graph centrality algorithms to identify the most influential KPIs that correlate with an observed performance anomaly. Other statistical approaches, like ϵ-Diagnosis (Shan et al. 2019) use statistical tests to rank metrics based on how much their distribution changes during a failure.

While correlation is useful, it does not imply causation. More advanced methods leverage causal inference or related techniques to build a more accurate picture of fault propagation. A key step in this direction is accurately identifying when a system's behavior changes. BARO (Pham et al. 2024) proposes an end-to-end framework that first uses Multivariate Bayesian Online Change Point Detection to precisely identify the failure's start time. It then uses a robust scoring method based on the median and interquartile range–which are less sensitive to outliers and inexact timestamps–to rank potential root causes. CIRCA (Li et al. 2022) formulates RCA as an "intervention recognition" task. It constructs a causal graph using system architecture knowledge and then uses regression-based hypothesis testing to find the metric whose conditional probability distribution (given its parents) changes most significantly. In contrast, RCD (Root Cause Discovery) (Ikram et al. 2022) discovers causal relationships directly from data without requiring a predefined graph. It introduces a special "Failure Node" to represent the intervention and uses a highly efficient hierarchical and localized algorithm to find the direct causes of this node, avoiding the expensive process of building the full causal graph.

There are also many works focusing on searching fault-indicating attribute combinations of KPI data. CMMD (Yan et al. 2022) 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 (Zhang et al. 2021) proposed a hierarchical search approach to capture the relationship among attributes based on conditional entropy and locate the fault-indicating combination. Another approach iDice (Lin et al. 2016a) 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 (Gu et al. 2020) 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.

8 Conclusion

In this paper, we propose KPIRoot+ , an effective and efficient framework for anomaly detection and root cause analysis in practical cloud systems with monitoring KPIs. Specifically, KPIRoot+ is an improved version of KPIRoot that utilizes time decomposition-based anomaly detection and improved SAX representation, offering more accurate root cause localization results, while not compromising the efficiency. Extensive experiments on three industrial datasets show that KPIRoot achieves 0.882 F1-Score and 0.946 Hit@10 with the highest efficiency, outperforming all the baselines, including KPIRoot. Moreover, the successful deployment of our approach in large-scale industrial applications further demonstrates its practicality.

Author Contributions Data curation: Xinying Sun, Yongqiang Yang; Funding acquisition: Michael R. Lyu, Guangba Yu, Jiazhen Gu; Methodology: Wenwei Gu; Supervision: Michael R. Lyu; Validation: Renyi Zhong; Visualization: Jinyang Liu, Yintong Huo, Zhuangbin Chen, Jianping Zhang; Original Draft: Wenwei Gu; Review: Guangba Yu, Jiazhen Gu.

Funding 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).

Data Availability The full data cannot be made available due to the privacy policy in Cloud $\mathcal{H}$. Only a portion of desensitized samples will be made public. The code is released in: https://github.com/WenweiGu/KPIRoot+.

Declarations

Conflicts of Interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethical Approval This manuscript extends our previous ISSRE paper titled ’KPIRoot: Efficient Monitoring Metric-based Root Cause Localization in Large-scale Cloud Systems,’ which further improves the KPIRoot. The authors also declare that this manuscript follows the best scientific standards, in particular with regard to acknowledgment of prior works, honesty of the presentation of results, and focus on the demonstrability of the statements. This manuscript and the work that led to it do not carry any specific ethical issue.

Informed Consent All the authors give their consent to submit this work.

References

Agarwal S, Chakraborty S, Garg S, et al (2023) Outage-watch: Early prediction of outages using extreme event regularizer. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp 682–694

Arnold A, Liu Y, Abe N (2007) Temporal causal modeling with graphical granger methods. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 66–75

Beloglazov A, Buyya R (2012) 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 24(13):1397–1420

Breunig MM, Kriegel HP, Ng RT, et al (2000) Lof: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pp 93–104

Chen Y, Xu D, Chen N et al (2023) Frl-mfpg: Propagation-aware fault root cause location for microservice intelligent operation and maintenance. Inf Softw Technol 153:107083

Cheng Q, Sahoo D, Saha A, et al (2023) Ai for it operations (aiops) on cloud platforms: Reviews, opportunities and challenges. arXiv preprint arXiv:2304.04661

Cortez E, Bonde A, Muzio A, et al (2017) Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp 153–167

Deng A, Hooi B (2021) Graph neural network-based anomaly detection in multivariate time series. In: Proceedings of the AAAI conference on artificial intelligence, pp 4027–4035

Gu J, Luo C, Qin S, et al (2020) 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, pp 292–303

Gu W, Liu J, Chen Z, et al (2024a) Identifying performance issues in cloud service systems based on relational-temporal features. ACM Transactions on Software Engineering and Methodology

Gu W, Sun X, Liu J, et al (2024b) Kpiroot: Efficient monitoring metric-based root cause localization in large-scale cloud systems. In: 2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE), IEEE, pp 403–414

Guo C, Li H, Pan D (2010) 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, pp 234–244

He X, Shao C, Xiong Y (2016) A non-parametric symbolic approximate representation for long time series. Pattern Anal Appl 19:111–127

Huang T, Chen P, Zhang J, et al (2022) 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, pp 1–12

Ikram A, Chakraborty S, Mitra S et al (2022) Root cause analysis of failures in microservices through causal discovery. Adv Neural Inf Process Syst 35:31158–31170

Kaushik P, Rao AM, Singh DP, et al (2021) 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, pp 268–273

Kuang J, Liu J, Huang J, et al (2024) 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, pp 369–380

Latah M, Toker L (2019) Artificial intelligence enabled software-defined networking: a comprehensive overview. IET networks 8(2):79–99

Li M, Li Z, Yin K, et al (2022) Causal inference-based root cause analysis for online service systems with intervention recognition. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp 3230–3240

Li P, Pei Y, Li J (2023) A comprehensive survey on design and application of autoencoder in deep learning. Appl Soft Comput 138:110176

Lin J, Keogh E, Lonardi S, et al (2003) 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, pp 2–11

Lin Q, Lou JG, Zhang H, et al (2016a) idice: problem identification for emerging issues. In: Proceedings of the 38th International Conference on Software Engineering, pp 214–224

Lin Q, Zhang H, Lou JG, et al (2016b) Log clustering based problem identification for online service systems. In: Proceedings of the 38th International Conference on Software Engineering Companion, pp 102–111

Lin Q, Hsieh K, Dang Y, et al (2018) 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, pp 480–490

Liu FT, Ting KM, Zhou ZH (2008) Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining (ICDM), IEEE, pp 413–422

Liu J, Wang S, Zhou A et al (2016) Using proactive fault-tolerance approach to enhance cloud service reliability. IEEE Transactions on Cloud Computing 6(4):1191–1202

Liu J, He S, Chen Z, et al (2023) Incident-aware duplicate ticket aggregation for cloud systems. arXiv preprint arXiv:2302.09520

Ma M, Lin W, Pan D et al (2021) Servicerank: Root cause identification of anomaly in large-scale microservice architectures. IEEE Trans Dependable Secure Comput 19(5):3087–3100

Mariani L, Monni C, Pezzé M et al (2018) Localizing faults in cloud systems. 2018 IEEE 11th International Conference on Software Testing. Verification and Validation (ICST), IEEE, pp 262–273

Mariani L, Pezzè M, Riganelli O et al (2020) Predicting failures in multi-tier distributed systems. J Syst Softw 161:110464

Minnen D, Isbell C, Essa I, et al (2007) Detecting subdimensional motifs: An efficient algorithm for generalized multivariate pattern discovery. In: Seventh IEEE International Conference on Data Mining (ICDM 2007), IEEE, pp 601–606

Pham L, Ha H, Zhang H (2024) Baro: Robust root cause analysis for microservices via multivariate bayesian online change point detection. Proceedings of the ACM on Software Engineering 1(FSE):2214–2237

Qiu J, Du Q, Yin K et al (2020) A causality mining and knowledge graph-based method of root cause diagnosis for performance anomaly in cloud applications. Appl Sci 10(6):2166

RB C (1990) Stl: A seasonal-trend decomposition procedure based on loess. J Off Stat 6:3–73

Ren H, Xu B, Wang Y, et al (2019) Time-series anomaly detection service at microsoft. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp 3009–3017

Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18, Springer, pp 234–241

Senin P, Malinchik S (2013) Sax-vsm: Interpretable time series classification using sax and vector space model. In: 2013 IEEE 13th international conference on data mining, IEEE, pp 1175–1180

Shan H, Chen Y, Liu H, et al (2019) ϵ-diagnosis: Unsupervised and real-time diagnosis of small-window long-tail latency in large-scale microservice platforms. In: The World Wide Web Conference, pp 3215–3222

Sharma Y, Bhamare D, Sastry N, et al (2023) Sla management in intent-driven service management systems: A taxonomy and future directions. ACM Computing Surveys

Shojaie A, Fox EB (2022) Granger causality: A review and recent advances. Annual Review of Statistics and Its Application 9:289–319

Siffer A, Fouque PA, Termier A, et al (2017) Anomaly detection in streams with extreme value theory. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp 1067–1075

Singh S, Batheri R, Dias J (2023) Predictive analytics: How to improve availability of manufacturing equipment in automotive firms. IEEE Engineering Management Review

Soldani J, Brogi A (2022) Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey. ACM Computing Surveys (CSUR) 55(3):1–39

Su Y, Zhao Y, Niu C, et al (2019a) 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, pp 2828–2837

Su Y, Zhao Y, Xia W, et al (2019b) Coflux: robustly correlating kpis by fluctuations for service troubleshooting. In: Proceedings of the International Symposium on Quality of Service, pp 1–10

Tuli S, Gill SS, Garraghan P, et al (2021) Start: Straggler prediction and mitigation for cloud computing environments using encoder lstm networks. IEEE Transactions on Services Computing

Tuli S, Casale G, Jennings NR (2022) Tranad: deep transformer networks for anomaly detection in multivariate time series data. Proceedings of the VLDB Endowment 15(6):1201–1214

Wang H, Nguyen P, Li J et al (2019) Grano: Interactive graph-based root cause analysis for cloud-native distributed data platform. Proceedings of the VLDB Endowment 12(12):1942–1945

Wang D, Chen Z, Ni J, et al (2023a) Hierarchical graph neural networks for causal discovery and root cause localization. arXiv preprint arXiv:2302.01987

Wang L, Zhang C, Ding R, et al (2023b) 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, pp 5116–5125

Wickremasinghe B, Calheiros RN, Buyya R (2010) 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, pp 446–452

Wu L, Tordsson J, Bogatinovski J, et al (2021) Microdiag: Fine-grained performance diagnosis for microservice systems. In: 2021 IEEE/ACM International Workshop on Cloud Intelligence (CloudIntelligence), IEEE, pp 31–36

Xu H, Chen W, Zhao N, et al (2018) Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In: Proceedings of the 2018 World Wide Web Conference, pp 187–196

Yan S, Shan C, Yang W, et al (2022) Cmmd: Cross-metric multi-dimensional root cause analysis. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp 4310–4320

Yang T, Shen J, Su Y, et al (2021) 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, pp 653–665

Yin J, Zhao X, Tang Y et al (2016) Cloudscout: A non-intrusive approach to service dependency discovery. IEEE Trans Parallel Distrib Syst 28(5):1271–1284

Yu B, Yao J, Fu Q, et al (2024) 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, pp 1–13

Yu G, Chen P, Li Y, et al (2023) 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, pp 553–565

Zhang J, Wu W, Huang Jt, et al (2022) Improving adversarial transferability via neuron attribution-based attacks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 14993–15002

Zhang J, Gu W, Huang Y, et al (2024) Curvature-invariant adversarial attacks for 3d point clouds. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp 7142–7150

Zhang T, Ramakrishnan R, Livny M (1996) Birch: an efficient data clustering method for very large databases. ACM SIGMOD Rec 25(2):103–114

Zhang X, Du C, Li Y, et al (2021) Halo: Hierarchy-aware fault localization for cloud systems. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp 3948–3958

Zhao N, Zhu J, Liu R, et al (2019) Label-less: A semi-automatic labelling tool for kpi anomalies. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, pp 1882–1890

Zhao N, Chen J, Peng X, et al (2020a) 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, pp 162–171

Zhao N, Chen J, Wang Z, et al (2020b) 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, pp 315–326

Zhou F, Goel M, Desnoyers P et al (2013) Scheduler vulnerabilities and coordinated attacks in cloud computing. J Comput Secur 21(4):533–559

Zong B, Song Q, Min MR, et al (2018) Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In: International conference on learning representations

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Wenwei Gu[1] · Renyi Zhong[1] · Guangba Yu[1] · Xinying Sun[4] · Jinyang Liu[1] · Yintong Huo[3] · Zhuangbin Chen[2] · Jianping Zhang[1] · Jiazhen Gu[1] · Yongqiang Yang[4] · Michael R. Lyu[1]

✉ Guangba Yu
guangbayu@cuhk.edu.hk

Wenwei Gu
wwgu21@cse.cuhk.edu.hk

Renyi Zhong
ryzhong22@cse.cuhk.edu.hk

Xinying Sun
sunxinying1@huawei.com

Jinyang Liu
jyliu@cse.cuhk.edu.hk

Yintong Huo
ythuo@smu.edu.sg

Zhuangbin Chen
chenzhb36@mail.sysu.edu.cn

Jianping Zhang
jpzhang@cse.cuhk.edu.hk

Jiazhen Gu
jiazhengu@cuhk.edu.hk

Yongqiang Yang
yangyongqiang@huawei.com

Michael R. Lyu
lyu@cse.cuhk.edu.hk

[1] The Chinese University of Hong Kong, Ma Liu Shui, Hong Kong SAR

[2] Sun Yat-sen University, Guangzhou, China

[3] Singapore Management University, Singapore, Singapore

[4] Huawei Cloud Computing Technology Co., Ltd, Shenzhen, China