A Metric Recommendation Service for Online Systems using Graph Learning

Anonymous Author(s)*

***Abstract*—Today's monitoring and failure management mechanisms for online systems heavily rely on metrics, which are time series data that can describe the real-time state of a system from various perspectives. Though several attempts have been devoted to automatic failure management based on metrics, the primary step, metric selection, remains manual to a large extent. To better understand the prior practice, we conduct an empirical study on the selected metrics in prior work and obtain some findings. Based on the findings, we develop a metric recommendation service for online systems, which can automate the metrics selection practice and greatly ease the burden in managing an online system. Specifically, we analyze the needs of two key failure management tasks, i.e., anomaly detection and fault diagnosis, and design metric recommendation mechanisms for them respectively. Graph learning techniques are employed in the automation of metric recommendation. Our experiments demonstrate that the proposed approach can achieve an F1-score of 0.912 in selecting metrics for anomaly detection, and an accuracy of 0.859 in retrieving metrics for faults diagnosis, which significantly outperforms the compared baselines.**

***Index Terms*—metric recommendation, graph learning, failure management, online systems**

I. Introduction

Given that modern online systems are gradually complex and large-scale, continually managing failures in these systems to ensure the high availability becomes increasingly important yet difficult. Several failure management tasks, e.g., anomaly detection [1]–[16] and fault diagnosis [17]–[25], are developed on the basis of metrics, e.g., Page View Count and CPU Usage. Metrics are time series data that record the real-time state of a system. Due to the explosively growing number of metrics that can be collected, metrics selection usually needs to be performed on demand to accelerate and enhance the management process. For example, when a software engineer wants to confirm whether the system is working properly, he/she shall first select some metrics and then perform anomaly detection on them. The selected metrics are usually referred to as *Key Performance Indicators (KPIs)*[1] since they are thought to well characterize the system and have a powerful ability in revealing faults. Another example is, when a software engineer is diagnosing a fault, he/she usually needs to pick out relevant metrics to better understand what is going on. In these cases, the selected metrics are the core clues in failure management. Therefore, their effectiveness is of critical importance.

Although various intelligent techniques have been applied to automatic anomaly detection and fault diagnosis for online systems, the common practice of metric selection remains manual yet. Typically, software engineers create and customize KPI dashboards to aid their daily operations. A KPI dashboard is a group of selected KPIs that are organized together in a Web-hosted site. Many failure management tasks, e.g., alert rule setting and fault diagnosis, are further performed on the basis of the configured KPI dashboards.

However, such manual practice has several limitations:

- **Low Scalability for Large-Scale Monitoring.** Modern online systems can be incredibly large-scale, with millions or even billions of metrics that can be collected from different entities. For examples, in an online system in Tencent, the volume of metric data can be over 500TB a day. Moreover, this number is still growing with the development of the system obeservabilty techniques [26]–[29]. Requiring software engineers to manually inspect all these data to find useful metrics is impractical.
- **Limited Flexibility for Evolving Systems.** Modern online systems are under rapid development. New functionalities are continually added to an online system, and the available metrics are continually increasing. Moreover, the usefulness of some metrics might change as the system evolves. This yields a need for software engineers to frequently update their KPI dashboards, which can be especially exhausting.
- **Heavy Reliance on Expert Knowledge.** Understanding which metrics are more useful can be challenging. Even experienced engineers can find it hard to distinguish the overwhelming amount of metrics today as the monitoring can be more and more fine-grained. Moreover, the appropriate metrics might vary against actual conditions. Therefore, it is unrealistic to assume that every engineer is experienced enough to select metrics manually.

Metric recommendation can help a lot. Specifically, if a metric recommendation service is provided, software engineers can simply trigger the service and set their KPI dashboards according to the recommendation. As the annoying procedures are automated, the burden of software engineers can be greatly alleviated. Moreover, various downstream tasks, e.g., anomaly detection and fault diagnosis, can benefit from this automatic process. In this paper, we focus on metric recommendation for failure management, including anomaly detection and fault diagnosis.

Some prior work [30]–[32] has recommended some metrics as KPIs. For example, Google [32] recommends using the latency, traffic pressure, error count and saturation as KPIs.

[1]It is noted that there is a clear distinction between these two concepts, i.e., metrics and KPIs, in this paper. KPIs only refer to the selected metrics for monitoring and anomaly detection.

Besides, there exist some open-sourced KPI dashboards [33] recommended to software engineers. Following these practices is useful in some scenarios. However, these recommendations are static. As a result, their recommended KPIs might not be always appropriate nor enough, especially considering that modern online systems are under rapid development and the possible faults are diverse. Moreover, there are too many static recommendations in the wild. As a result, the selection of these static recommendations is back to a manual task, which is still very difficult as discussed above. We will further systematically investigate these practices in Section III.

There also exist some prior work [34]–[36] which automatically selects metrics to perform anomaly detection for systems. However, they require continuous labelling of the anomaly time to supervisedly learn the correlation between metrics and real faults, which is still very labor-intensive and difficult to apply in practice. Moreover, they only target at one failure management task, i.e., the anomaly detection, rendering their usage limited.

Compared with the practices discussed above, this paper proposes to automatically recommend metrics against actual conditions with minimal human effort. We analyze the need of different failure management tasks, including anomaly detection and fault diagnosis, and design a data-driven metric recommendation method which can be easily applied across systems. This can pose several challenges: i) How to accommodate the need of different failure management tasks when performing metric recommendation? ii) How to strike a balance between the effectiveness and the reliance on human effort during the recommendation process?

To address the above challenges, we propose a metric recommendation service for online systems using graph learning. According to the different needs of failure management in practice, we decompose the metric recommendation service into two usage scenarios, namely metric selection for anomaly detection and unmonitored anomaly-related metric retrieval for fault diagnosis. Then, graph learning techniques including random walk and graph neural network are employed to overcome the challenges discussed above. In general, our contributions are as follows,

- We conduct the first empirical study on the metric selection practices in the wild, and gain a series of findings.
- We propose a metric recommendation service for online systems based on graph learning, which can greatly automate metric selection for anomaly detection and metric retrieval for fault diagnosis.
- We conduct extensive experiments to validate the proposed approach on metric data collected from Tencent®. The results confirm the effectiveness of the proposed approach in recommending metrics for failure management in online systems.

II. Background

In this section, we introduce the background of metric recommendation for online systems in practice, including the basic concepts about a KPI dashboard, the metrics in online systems and their usages for failure management.

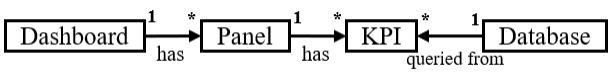

Fig. 1. The relationships of basic concepts about a KPI dashboard

A. KPI Dashboard

Inside a KPI dashboard. Fig. 1 shows the relationships of some basic concepts about a KPI dashboard. A KPI dashboard is the entry point for many failure management tasks. Software engineers usually use it to inspect the state of a system, or configure the inputs for some intelligent anomaly detection and fault diagnosis algorithms. A KPI dashboard consist of several panels, and each panel is configured with some KPIs to show specific real-time properties of a system. Typically, KPIs in the same panel are metrics of the same type but with different attributes. For example, the CPU Usages (metric type) of different servers (metric attributes) are usually placed in the same panel. Another example is the request number (metric type) of different response status (metric attributes). However, which metric types and which attributes should be selected as KPIs for display and anomaly detection still remain a question. Therefore, in this paper, we provide a mechanism to automatically recommend KPIs for anomaly detection on the basis of the historical data of different metrics.

Outside a KPI dashboard. Typically, the KPIs are queried from a remote database. Since the attention of a software engineer is limited, he/she usually can only place a few KPIs in a dashboard for analysis. Despite the displayed KPIs, there are still many metrics that can be queried from the database. Since these metrics are usually not used by any software engineers, we call them *unmonitored metrics* in this paper. Although they are neglected in a KPI dashboard, they may still be useful for the diagnosis of some specific faults. Therefore, in this paper, we also propose a method to recommend these metrics during the diagnosis of a fault.

B. Metrics in Online Systems

Varieties of metrics. There are various metrics that can be collected in modern online systems. We can roughly classify them into three categories, i.e., resource-related metrics, component-specific metrics and business-level metrics.

Resource-related metrics denote metrics that describe the resource including the CPU, the memory, the disk storage, the network, the process and so on. As the system obeservabilty techniques [26]–[29] develop in recent years, more and more fine-grained resource-related metrics can be collected, which provide insightful details of the system.

Component-specific metrics refer to metrics that describe some software components deployed as part of the online system, e.g., the database (MySQL, MongoDB, Redis and so on) and the messaging systems (Kafka, RabbitMQ and so on). For these metrics, requiring software engineers to manually select the most appropriate metrics to monitor every component is labor-intensive and error-prone.

Business-level metrics are metrics that describe the quality of the business, e.g., the advertisement revenue and the number of online users. These metrics are usually customized by the engineers, and are usually KPIs by default. However, only relying on business-level metrics to perform anomaly detection is not enough, because it is too late if we only react to a fault when the business is seriously damaged. Besides, fault diagnosis heavy relies on resource-related metrics and component-specific metrics.

Attributes of metrics. Apart from the variety of metric types, there might be numerous attributes associated with some metrics, which should be configured when querying the metrics. In some scenarios, the configuration of some attributes has a great impact on the effectiveness of the selected metrics. This further increase the difficulty in selecting right metrics with appropriate attributes.

C. Using Metrics for Failure Management

In this section, we describe different usages of KPIs for failure management in practice, which can shed light on the design of our KPI recommendation solution.

Monitoring and Anomaly Detection. The initial intent for creating a KPI dashboard is to monitor some properties of a running system and build confidence on it. For example, one might need to confirm whether the system is working properly after he/she updates some configurations of the system. Apart from manual monitoring, some statistical or artificial intelligence techniques [1]–[16] can be applied to automatic anomaly detection on metrics, which greatly improves the operational efficiency. Since the effectiveness of the input metrics greatly affect the accuracy of the anomaly detection algorithms, metric recommendation in advance can help a lot.

Fault Diagnosis. If some problems occur, the fault diagnosis heavy relies on metrics too since some metrics can provide clues about the root cause of a fault. Fault diagnosis is to find these metrics. This tasks can be conducted manually, or performed by some artificial intelligence techniques developed recently [17]–[25]. However, no matter how this task is performed, selecting useful metrics to assess the symptom of a fault needs to be conducted first. It is noteworthy that the selected metrics here might not be KPIs since the possible faults are diverse, and the metrics for fault diagnosis should be selected against actual conditions. For this usage, metric recommendation plays a vital role.

III. Empirical Study on Metric Selection Practice

In this section, we investigate the current practice in selecting metrics to aid failure management, and aim to answer the following questions.

- **RQ1:** How is the current practice in selecting metrics on KPI dashboards for monitoring?
- **RQ2:** How is the current practice in selecting metrics for intelligent anomaly detection and fault diagnosis in online systems?

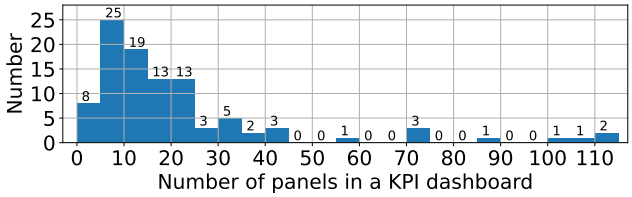

Fig. 2. The histogram of the number of panels in the collected KPI dashboards.

A. Subjects

For RQ1, we collect 100 open-sourced KPI dashboards with the top downloads (each downloaded by 171 thousands $\sim$ 18 millions users) from Grafana [33]. Grafana is a top KPI dashboard engine, which is representative enough to reveal the practice of engineers in selecting metrics on KPI dashboards for monitoring.

For RQ2, we study the metric selection practice in the state-of-the-art work on intelligent anomaly detection and fault diagnosis for online systems. We search for the relevant work from some top conferences or journals on software engineering or data mining in recent 3 years. Then, work that contains a clear description of the selected metrics is discussed here.

B. RQ1: KPI Dashboard Analysis

As introduced in Section II-A, each panel in a KPI dashboard usually contain metrics of the same type. For instance, an engineer usually tend to place CPU usage in one panel, and then place memory usage in another panel. Therefore, we first calculate the number of panels in each KPI dashboard to investigate how many metrics will usually be displayed for the monitoring of an entity (e.g., a machine, or a software component such as database, which depends on the target of the corresponding dashboard) in practice. Fig. 2 displays a histogram summarizing the number of panels in each dashboard. It can be observed that more than three quarters of KPI dashboards have less than 25 metrics. This is reasonable because only displaying a few key metrics in a dashboard can actually ease the burden of monitoring the system.

C. RQ2: Intelligent Failure Management Method Analysis

Table I displays the statistics of metrics used by some representative intelligent failure management methods for online systems. It can be observed that even when the processing is automatic, engineers will not input all the collectable metrics (the number of metrics that can be collected is far larger than the number shown in the table) into the anomaly detection algorithms. This is reasonable because many metrics are noisy or contain redundant information. Therefore, selecting KPIs for anomaly detection is a common practice. However, as we can see in the third column in the table, the KPIs selected in different systems are still different, which are mainly determined by corresponding experts maintaining the systems. Moreover, such a practice is not automatic, and there is still no a unified standard for selecting KPIs, rendering it difficult to release the full potential of these anomaly detection methods in different systems.

TABLE I
THE STATISTICS OF METRICS USED BY SOME REPRESENTATIVE INTELLIGENT FAILURE MANAGEMENT METHODS IN RECENT 3 YEARS

Failure Management Tasks	Representative Methods	Number of Selected Metrics	Categories of Selected Metrics
Anomaly Detection	OmniAnomaly [9]	38	Resource-related metrics
	TopoMAD [15]	7∼26	Resource-related metrics
	JumpStarter [2]	19	Resource-related metrics
	CTF [14]	49	Resource-related metrics
	SCWarn [16]	8	Resource-related metrics
Fault Diagnosis	MicroCause [24]	68	Resource-related metrics, component-specific metrics
	PatternMiner [25]	265∼5213	Resource-related metrics, component-specific metrics
	Dejavu [17]	120	Resource-related metrics, component-specific metrics

As for intelligent fault diagnosis, we can see that engineers tend to use more metrics for analysis. This is reasonable because possible faults are diverse. It is difficult to specify a scope of faults and guarantee that only the specified faults may happen in the system. Since more available metrics can provide more clues, it is better to take more metrics into consideration when developing an intelligent fault diagnosis method.

D. Summary

In general, our findings can be summarized as follows:

- Engineers tend to select only a few metrics for monitoring (i.e., typically less than 25 metrics in a KPI dashboard) and intelligent anomaly detection (i.e., less than 50 metrics as inputs of an algorithm).
- There is still no a unified standard for selecting metrics for intelligent anomaly detection and fault diagnosis.
- Fault diagnosis tends to require more metrics than anomaly detection, and thus the candidate metrics for fault diagnosis should not be limited to KPIs.

IV. MOTIVATION

A. Problem Formulation

For different purposes, the useful metrics varies. This paper proposes to recommend metrics against actual conditions, and targets at making full use of metrics that can be collected, not limited to KPIs. The following contains the usage scenario formulations of metric recommendation for failure management.

Metric Selection for Anomaly Detection. In an online system, metric data are collected from enormous entities (e.g., machines, services), and then stored into a database. They can be fetched to a dashboard for monitoring, or further inputted to some machine learning models for anomaly detection. As described in Section II-B, each entity can be associated with many metrics. As a result, the primary problem is, *"For each entity, which metrics can better reveal the entity state and should be utilized for anomaly detection?"*. Specifically, for an entity, a set of metrics $\mathbf{M} = \{M_1, M_2, \cdots\}$ can be collected, with a metric $M_i = [x_i^{t-w_s}, \cdots, x_i^t]$, where x_i^t is its observation at time t and w_s is the observation window size. The objective is to gain a recommended subset of $\mathbf{M}$, and the recommended subset should comply with the fault revealing goal, noise removal goal and pattern diversity goal, which will be further discussed in Section IV-B.

Unmonitored Anomaly-Related Metric Retrieval for Fault Diagnosis. Apart from the selected KPIs for anomaly detection, there may still be some unmonitored metrics which are helpful for the diagnosis of some faults. Previously, these metrics may be simply neglected by the engineers. This paper aims at making the full use of metrics that can be collected, and proposes to retrieve the unmonitored metrics on demand for fault diagnosis. Therefore, the problem is, *"Given an anomalous KPI, which unmonitored metrics may expose some clues about the anomaly and need to be retrieved for analysis?"*. Specifically, given an anomalous KPI M^a and a set of other unmonitored metric $\mathbf{M}^u = \{M_1^u, M_2^u, \cdots\}$, the objective is to calculate a recommended subset of $\mathbf{M}^u$ to retrieve for fault diagnosis. The recommended subset should comply with the fault understanding goal, which will be discussed in Section IV-B.

B. Metric Recommendation Goals

Based on the introduction of metric usages in Section II-C and the problem formulation described above, we summarize the goals of metric recommendation as follows.

1) Common Goal: **Effort Reducing Goal.** The proposed metric recommendation method is to reduce human effort. Therefore, we should not require engineers to continuously provide some external inputs (e.g., the time of all the historical faults for a supervised learning) to gain the recommendation. Even when such external inputs are a must, the effort should be one-time, and reusable by the same system over time as well as other systems.

2) Metric Selection Goals: The followings are some specific goals for metric selection.

Fault Revealing Goal. If a fault occurs, the selected KPIs should have a high probability to behave abnormally. Otherwise, many faults may be neglected, and thus the KPIs are actually useless. Therefore, one goal of metric selection is to select KPIs that can successfully reveal faults in a system.

Noise Removal Goal. Some metrics are noisy and using them for alerting might give rise to a flood of false alerts. Therefore, one goal of metric recommendation is to exclude these noisy metrics.

Pattern Diversity Goal. In an online system, some metrics might correlate with each other. Within the same entity, correlated metrics contain redundant information, and should be carefully handled. Therefore, one goal of metrics selection is to identify and handle this redundancy and then promote the pattern diversity in the recommended metrics.

3) Metric Retrieval Goal: The following is the specific goal for metric retrieval.

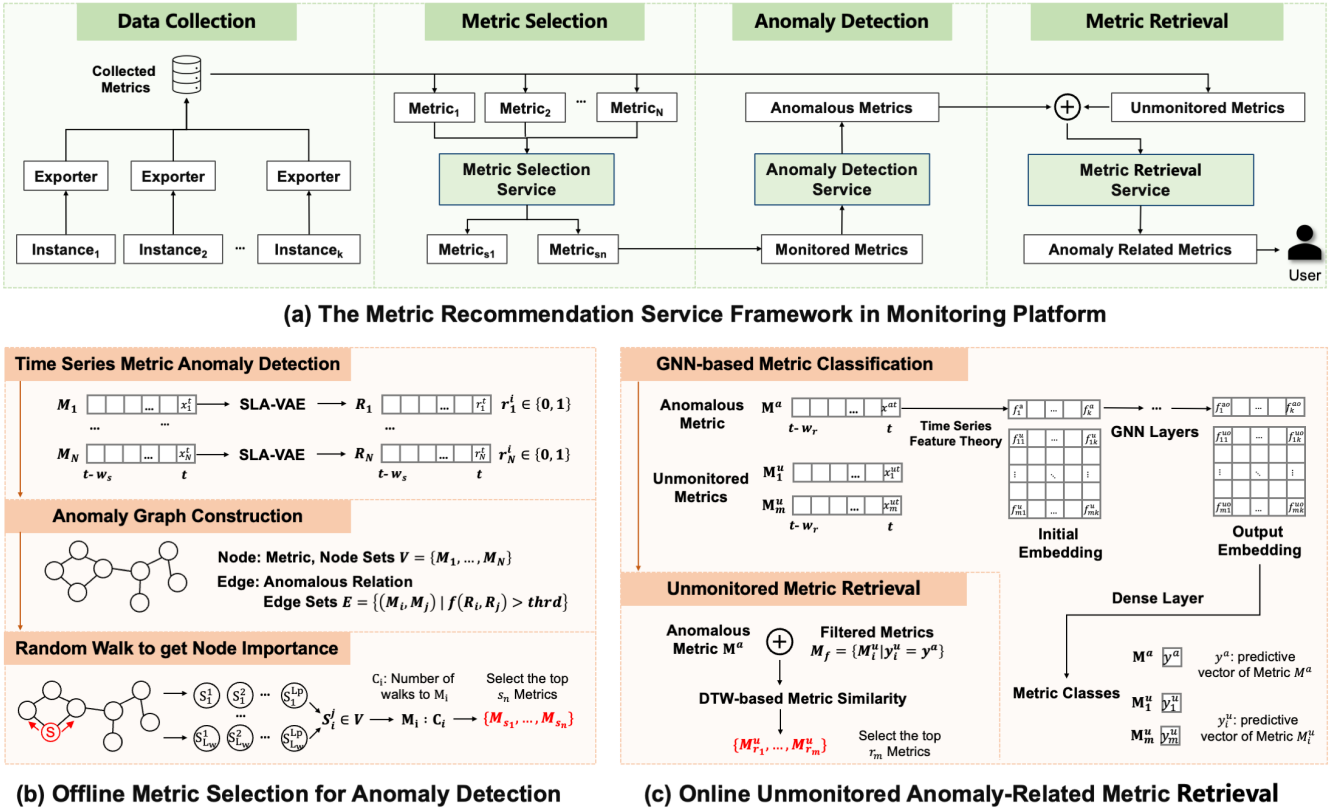

Fig. 3. The metric recommendation service framework in monitoring platform, and the metric selection and retrieval methodologies.

Fault Understanding Goal. Diagnosing a fault requires an in-depth investigation into the impact of the fault. Therefore, the goal of metric retrieval is to retrieve metrics that can provide a comprehensive view of a fault.

V. Methodology

To address the practical limitations and accomplish the metric recommendation goals, we propose a metric recommendation service including offline metric selection for anomaly detection and online unmonitored anomaly-related metric retrieval for fault diagnosis. Fig. 3 presents the metric recommendation service framework in monitoring platform, and the metric selection and retrieval methodologies. In this section, we will elaborate on the methodology design of the offline metric selection and online unmonitored anomaly-related metric retrieval.

A. Overview of Metric Recommendation Service

To ensure the availability of online systems, software engineers need to collect metrics from numerous online system instances, select some of these metrics to monitor based on their expert knowledge, and then adopt anomaly detection techniques to identify anomalies in these metrics. When there exist anomalies, the software engineers should diagnose and repair the system faults based on the anomalous metrics. However, it is difficult to ensure that metrics can be selected completely and accurately. The engineers also need a service to assist them in automatically selecting metrics that can characterize the availability of their systems. Hence, we first design a metric selection method for monitoring and detecting anomalies. In addition, the unmonitored metrics can also assist in revealing and diagnosing anomalies. There are far more unmonitored metrics than monitored metrics in an industrial environment. Adopting only monitored metrics may make it difficult to diagnose a fault. Therefore, we design an anomaly-related unmonitored metric retrieval method for fault diagnosis.

Fig. 3(a) shows the framework of the proposed metric recommendation service in monitoring platform. The metrics are first collected from the system instances and stored in database. Then, we adopt the metric selection service to recommend KPIs which should be monitored. Software engineers can manually integrate the recommendation results and their expert knowledge to build relevant monitoring tasks. We can also automatically build the monitoring tasks based on the recommendation results. Next, we use an anomaly detection service to identify anomalies. Since the anomaly detection technique is not the focus of this paper, we will not elaborate on the anomaly detection algorithm. In this paper, we adopt semi-supervised variational auto-encoder (SLA-VAE) [3] to detect anomalies in metrics. Eventually, we use the metric retrieval service to identify the unmonitored anomaly-related metrics. When monitored metrics are detected as anomalies,

the unmonitored anomaly-related metrics will also be sent to software engineers for further fault diagnosis.

B. Metric Selection for Anomaly Detection

The selected metrics, namely Key Performance Indicators (KPIs), are required to reveal the system faults effectively. When the system fails, the monitored KPIs should be anomalous so that the software engineers can notice and repair the fault timely. We may adopt the correlation between metric anomalies and system faults to automatic select metric in a supervised way, just as prior work [34], [36] does. However, there are two issues requiring us to design a new unsupervised solution in practice. First, the sample labels about system faults are usually incomplete and even lost in industrial environments. If the samples do not contain enough information about system faults, it is difficult for the recommended KPIs to detect all system failures. Second, due to the differences between systems, we should select different KPIs which are suitable for different systems. When we adopt a supervised method to select metrics, we need to collect and generate different samples for each system, which is disadvantageous for the wide deployment of the metric selection method.

To better design the unsupervised metric selection method, we further deconstruct the target problem. If the anomaly detection results of one metric are similar to the results of others metrics, it means that this metric can also identify the anomalies of other metrics and we can select this metric as KPI. Then the correlation between metric anomalies and system faults can be transformed into the relationships between metric anomalies. Eventually, we can construct the anomaly graph of metrics based on the metric anomaly relationships and select the KPIs via graph learning.

Fig. 3(b) shows the structure of the proposed unsupervised **M**etric **S**election approach using **G**raph learning (MSG). Since we focus on metric anomaly relationships, we first adopt a time series metric anomaly detection method to transform the original metrics into detection results. Then we adopt the detection results to construct the graph which can represent the topological relationships between metrics. Finally, we use graph representation learning techniques to learn the metric importance and select the representative KPIs.

1) Time series metric anomaly detection: The metric set to be selected can be defined as $\mathbf{M} = \{M_1, M_2, \cdots, M_N\}$, where N is the number of collected metrics, $M_i = [x_i^{t-w_s}, \cdots, x_i^t]$ is the i^{th} metric which contains collected observations from time $t - w_s$ to t, and w_s is the observation window size. Time series metric anomaly detection aims to determine whether the observations are anomalous or normal. When identifying anomalies, the anomaly detection technique should be available to all kinds of metrics. In this paper, we adopt SLA-VAE [3] to identify anomalies. SLA-VAE contains an anomaly definition based feature extraction module, allowing it to be effective in the detection of different types of anomalies. Then the detection results of the metric set can be expressed as $\mathbf{R} = \{R_1, R_2, \cdots, R_N\}$, where $R_i = [r_i^{t-w_s}, \cdots, r_i^t]$ is the detection results of the i^{th} metric, and $r_i^k \in \{0, 1\}$ represents whether the observation of the i^{th} metric at time k is normal or anomalous.

2) Anomaly graph construction: The constructed anomaly graph of metric set can be defined as $G = (V, E)$, where $V = \{M_1, M_2, \cdots, M_N\}$ is the metric set and $E = \{(M_i, M_j) | f(R_i, R_j) > \tau\}$ is the anomalous relation set. The function f is adopted to measure the anomalous relation between metric M_i and M_j, and the threshold τ is used to filter the unrelated metric pair (M_i, M_j). When designing the function f, we transform the measure of anomalous relation among metrics into anomaly detection performance evaluation and adopt averaged F-measure [37] to characterize the anomalous relation. The function f can be expressed as:

$$f(R_i, R_j) = \frac{2 * p(R_i, R_j) * p(R_j, R_i)}{p(R_i, R_j) + p(R_j, R_i)} \tag{1}$$

$$p(R_i, R_j) = \frac{\#_{k \in [t-w_s, t]}(r_i^k = 1 \ \& \ r_j^k = 1)}{\#_{k \in [t-w_s, t]}(r_i^k = 1)} \tag{2}$$

where $\#_{k \in [t-w_s, t]}(r_i^k = 1)$ is the number of anomalies in the i^{th} metric M_i and $\#_{k \in [t-w_s, t]}(r_i^k = 1 \ \& \ r_j^k = 1)$ is the number of anomalies in both the i^{th} metric M_i and the j^{th} metric M_j in the meantime. In addition, when two metrics have similar time series shapes, it means that both metrics may reveal the same thing. Thus, we prune the anomaly graph based on the similarity between metrics and remove edges between metrics with similar shapes. In this paper, we adopt dynamic time warping (DTW) [38] algorithm to compute the similarity. If the DTW distance of two metrics is lower than a relevant threshold, we will remove the edge between them. Eventually, we will obtain a pruned anomaly graph which can characterize the anomaly relations between metrics.

3) Random walk to select metrics: Based on the anomaly graph, random walk algorithm [39], which is the classical algorithm for computing node embedding and importance in graph learning, is adopted to compute metric importance. The results of one round of random walk on the anomaly graph can be defined as $\{s_i^1, s_i^2, \cdots, s_i^{L_p} \mid s_i^j \in V\}$, where L_p is the path length of random walk. We repeatedly execute L_w rounds of random walks in parallel. In more detail, during each round of random walk, we first randomly selected one metric $s_i^1 \in V$ and then sample one of its neighbor metrics based on the anomalous relation. The sampling function is defined as:

$$P(s_i^j | s_i^{j-1}) = \frac{f(s_i^j, s_i^{j-1})}{\sum_{s_i^k \in V_n(s_i^{j-1})} f(s_i^k, s_i^{j-1})} \tag{3}$$

where $P(s_i^j | s_i^{j-1})$ is the probability that metric s_i^j will be sampled and $V_n(s_i^{j-1})$ is the neighbor metric set of s_i^{j-1} based on E. Besides, we should note that the neighbor metric set $V_n(s_i^j)$ does not contain metric s_i^{j-1}. If s_i^{j-1} has no neighbor metric or L_p metrics have been sampled, we will stop the current random walk. Eventually, we compute the number of walks to each metric and select the top s_n metrics $\{M_{s_1}, M_{s_2}, \cdots, M_{s_n}\}$ as KPIs.

C. Unmonitored Anomaly-related Metric Retrieval for Fault Diagnosis

The unmonitored anomaly-related metric retrieval aims to choose the metrics which represent the same implications as the anomalous metrics. This target can be transformed as the time series metric classification problem. We group metrics with similar implications into one category, and then adopt relevant classification approach to classify the metrics. The unmonitored metrics that fall into the same category as anomalous metrics can be understood as anomaly-related metrics. Hence, we design the metric retrieval approach based on metric classification. Fig. 3(c) shows structure of the proposed unmonitored anomaly-related **M**etric **R**etrieval approach using **G**raph neural network (MRG). We first adopt graph neural network (GNN) to classify metrics, and then use the DTW distance to recommend the top similar metrics.

1) GNN-based metric classification: The unmonitored metric set can be defined as $\mathbf{M}^u = \{M_1^u, M_2^u, \cdots, M_m^u\}$, where m is the number of unmonitored metrics, $M_i^u = [x_i^{u,t-w_r}, \cdots, x_i^{u,t}]$ is the i^{th} metric which includes collected observations from time $t - w_r$ to t, and w_r is the corresponding window size. The detected anomalous metric is $M^a = [x^{a,t-t_r}, \cdots, x^{a,t}]$. The GNN-based metric classification aims to classify the metrics $\mathbf{M}^u$ and M^a into user-designed categories $\mathbf{C} = \{c_1, c_2, \ldots, c_o\}$, where o is the number of categories.

First, we introduce time series feature theory to extract features as initial embedding input of graph layers. Comparing with adopting original observations as initial embedding, extracted features can not only retain the time series information to ensure performance, but also reduce the computational complexity of the graph neural network. Hence, we adopt feature extraction approaches in this paper. Previous approaches extract time series features from three perspectives, namely *statistical domain, temporal domain, and spectral domain* [40], [41]. Here we adopt TSFEL [41] to extract features from above three domains. Eventually, the extracted features of metric M_i^u can be defined as $e_i^u = [e_{i,1}^u, e_{i,2}^u, \cdots, e_{i,d}^u]$, where d is the number of features.

Second, we adopt graph neural network to leverage and process the topology information of metrics. Comparing with directly using the extracted features, GNN [42] utilizes both the time series information within metrics and the topology information between metrics for metric classification, which may result in better performance. In practice, there may involve tens of thousands of unmonitored metrics need to be recommended. We are required to optimize the process of aggregating neighbor metrics to ensure the efficiency of graph neural network. The aggregation process can be expressed as:

$$\hat{e}_i^{u,t+1} = \mathrm{AGG}_{t+1}(\{e_j^{u,t} | e_j^{u,t} \in V_m(e_i^{u,t})\}) \tag{4}$$

$$e_i^{u,t+1} = \sigma(W_{t+1} \cdot [e_i^{u,t+1} \| \hat{e}_i^{u,t+1}]) \tag{5}$$

where $e_i^{u,t}$ and $e_i^{u,t+1}$ are the embedding at layer t and $t+1$, $V_m(e_i^{u,t})$ is the embedding of neighbor metrics of metric M_i^u, AGG is the aggregation method for aggregating neighbor information, $\hat{e}_i^{u,t+1}$ is the aggregated information, and σ is the logistic sigmoid function. Our optimization for the aggregation process lies in the selection of $V_m(e_i^{u,t})$. Due to the large number of neighbor metrics, we randomly select metric samples from the whole metric set. Then we adopt euclidean distance to measure the similarity between $e_i^{u,t}$ and $e_j^{u,t}$ and select the top metrics with similar embedding as neighbor metrics. Finally, based on the neighbor metrics, the output embedding $e_i^u = [e_{i,1}^{u,o}, e_{i,2}^{u,o}, \cdots, e_{i,d}^{u,o}]$ can contain both the time series information within metrics and the topology information between metrics.

Third, we adopt another dense layer to process the output embedding of GNN and classify metrics. The output categories of unmonitored metrics can be denoted as $\hat{\mathbf{y}}^u = \{\hat{y}_1^u, \hat{y}_2^u, \cdots, \hat{y}_m^u\}$, where $\hat{y}_i^u$ is the output vector of unmonitored metric M_i^u. In addition, we train the graph layers and dense layer using cross entropy loss. The loss of metric M_i^u is denoted as:

$$\mathcal{L}_i = -\frac{1}{o}\sum_{j=1}^{o}[y_{i,j}^u \log \hat{y}_{i,j}^u + (1 - y_{i,j}^u)\log(1 - \hat{y}_{i,j}^u)] \tag{6}$$

where $\hat{y}_{i,j}^u$ is the predictive probability that M_i^u belongs to category c_j, and y_i^u is the ground truth vector. We choose the top predictive probability in the vector to determine the predictive category. Ultimately, the metric set $\mathbf{M}_f = \{M_i^u | y_i^u = y^a\}$, which are classified into the same category as M_a, is determined as target candidate set to be recommended.

2) Unmonitored anomaly-related metric retrieval: The target of unmonitored metric retrieval is to generate metric subset $\mathbf{M}_r^u = \{M_{q_1}^u, M_{q_2}^u, \cdots, M_{q_m}^u \mid M_{q_i}^u \in \mathbf{M}^u\}$ related to M_a, where r_m is the number of recommended metrics. Comparing with directly adopting $\mathbf{M}_f$ to reveal and localize anomalies, we still need to rank the target candidate set. In practical environment, there may exist hundreds or even thousands of metrics fall into the same category as M_a. The large number of unordered results will make it difficult for software engineers to make decisions. Hence, we further rank the candidate set and select the top q_m metrics for recommendation. In this paper, we adopt the DTW [38] algorithm to measure the similarity between M_a and metrics in the candidate set $\mathbf{M}_f$ and use this similarity to rank metrics. Eventually, we recommend the sorted anomaly-related metrics $\mathbf{M}_r^u$ to engineers for decision making.

VI. Evaluation

In this section, we will respectively evaluate whether the proposed metric selection method MSG contributes to anomaly detection and whether the metric retrieval approach MRG benefits fault localization.

A. *RQ3: How effective is metric selection for monitoring and anomaly detection?*

1) Experiment Design: To evaluate the performance of the proposed MSG, we design experiments by combining metric selection and anomaly detection. In practice, the generic exporters such as those in Prometheus [43] will collect more

TABLE II
STATISTICS OF EVALUATION SAMPLES FOR METRIC SELECTION

VM Number	Window Size	Total Points	Anomalous Points
400	7 days	4032000	328/0.008%

TABLE III
THE TOP 10 CONCERNED METRICS OF ENGINEERS

Metric Perspective	Metric Name	Metric Meaning
I/O Usage	io_util	average io usage
	io_avgrq_sz	average size of requests
CPU Usage	cpu_detail	CPU usage of single VM
	cpu_summary	average CPU usage
	cpu_detail_max	max CPU usage
Memory Usage	pct_used	used physical memory
	pct_usable	unused physical memory
	psc_pct_used	used psc memory
Disk Usage	disk_in_use	average disk usage
Network	speed_packets_recv	speed of packet receive

than fifty resource-related metrics for user to detect anomalies. TABLE I also shows the statistics of metrics used to detect anomalies. We can find that there are differences in the number and types of metrics selected in different industrial environments. In other words, there is no unified evaluation standard for selecting metrics. It is difficult to evaluate the performance of metric selection only from the metric level. We instead compare the anomaly detection performance using the selected metrics based on MSG with other baselines. In addition, one of the detection target that software engineers are most concerned about is cloud virtual machine (VM). Here we focus on evaluating whether the proposed method can effectively select metrics for detecting anomalies in cloud VM.

We collect the real-world metrics of cloud VM in Tencent® for evaluation. TABLE II shows the statistics of evaluation samples. We randomly select 400 cloud VMs. For each cloud VM, we collect all metrics with a window size w_s of 7 days, and the observation is collected with interval of one minute. Hence, there are totally $400 \times 7 \times 1440 = 4032000$ points to be detected. For each point, the exporter has collected 56 metrics from 9 perspectives which includes disk usage, i/o usage, memory usage, CPU usage, and so on. Taking disk usage as an example, the exporter collects three metrics, namely *disk_total*, *disk_used*, and *disk_in_use*. We use the multiple metrics to train anomaly detection models and detect whether the point is anomalous or not.

We use precision, recall, and F1-score [37] to evaluate the anomaly detection performance, where $\text{precision} = \frac{TP}{TP+FP}$, $\text{recall} = \frac{TP}{TP+FN}$, and F1-score$= \frac{2 \times precision \times recall}{precision+recall}$. We compare the performance difference between experimental group and baseline groups to verify the effectiveness of metric selection method. For experimental group, we use the metrics generated by our proposed MSG to train anomaly detection model. For baseline group, we consider the expert experience and random selection. Besides, we uniformly adopt SLA-VAE [3] as anomaly detection model in this paper.

2) Experiment results: We first compare the proposed MSG and expert experience (EE) to evaluate performance. To better understand the expert experience, we counted tens of thousands of related tasks running on monitoring platform. We also surveyed the main concerns of software engineers on monitoring cloud VM. For confidentiality, we only show the conclusion here without relevant statistics. TABLE III shows the top 10 concerned metrics of engineers which are selected as baseline metrics here. Besides, we adopt MSG to generate the top 10 metrics, which are presented in TABLE IV. Then we use the selected metrics to train model and detect anomalies respectively. The performance comparison results are shown in TABLE V. We can find that the anomaly detection performance based on MSG is significantly better than expert experience. The selected metric comparison and performance comparison can illustrate that metric selection based on expert experience may ignore important metrics that contribute to anomaly detection. In contrast, based on the anomalous behavior of metrics, MSG can assist in monitoring and detecting system anomalies comprehensively.

TABLE IV
THE TOP 10 SELECTED METRICS BASED ON MSG

Metric Perspective	Metric Name	Metric Meaning
I/O Usage	io_util	average io usage
CPU Usage	cpu_detail	CPU usage of single VM
Memory Usage	pct_used	used physical memory
	psc_pct_used	used psc memory
Disk Usage	disk_in_use	average disk usage
Network	speed_packets_recv	speed of packet receive
	speed_packets_send	speed of packet send
Env	proc_blocked_current	number of blocked proc
CPU load	load1	average load in 1min
	load_per_cpu	average load per cpu

TABLE V
THE PERFORMANCE COMPARISON BETWEEN MSG AND BASELINE

Method	Precision	Recall	F1-score
EE	0.779	0.706	0.741
MSG	**0.909**	**0.915**	**0.912**

We then focus on the impact of the number of metrics on anomaly detection performance. We randomly select $n = \{10, 20, 30, 40, 50\}$ metrics from the whole collected metrics to train model and detect anomalies, and compare the performance with MSG. Fig 4 presents the anomaly detection comparison results between MSG and randomly chosen baselines. We can find that using more metrics to detect anomalies will result in better performance when randomly selecting metrics. However, we only get relatively good results when selected metric set is close to the full set. It means that only some of the collected metrics are useful for anomaly detection. When these metrics are not selected, the detection performance is negatively affected. In addition, we also find that MSG outperforms baselines. It illustrates that MSG can correctly select the useful metrics for anomaly detection.

*B. **RQ4**: How effective is metric retrieval for recommending unmonitored anomaly-related metric?*

1) Experiment Design: When evaluating the performance of the proposed MRG, we mainly focus on the accuracy that whether the unmonitored metrics will be classified into the same category as those detected as anomalies. In practice, it

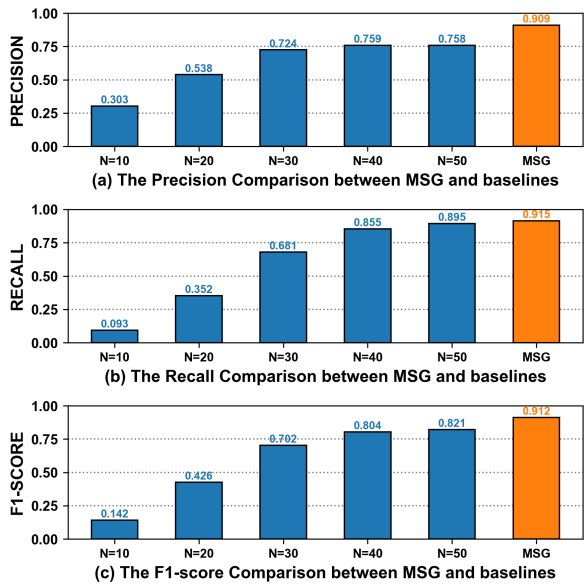

Fig. 4. The anomaly detection comparison results between MSG and randomly chosen baselines.

is difficult to establish and label the relationship between all unmonitored metric and anomalous metrics. To solve it, we transform the research question into time series classification problem as mentioned in section V-C. Both anomalous metrics and unmonitored metrics are adopted as input to MRG synchronously. The unmonitored metrics that fall into the same category as anomalous metrics can be understood as anomaly-related metrics. Therefore, we instead evaluate the accuracy that whether the metrics will be classified into the correct category.

We adopt the metrics collected from one system in Tencent® for evaluation. Table VI presents the statistics of evaluation sample for evaluating the proposed MRG. This evaluation sample contains all metrics collected from 400 cloud VMs, and the window size w_r of metrics is 2 days. Then we label the metrics from 8 label domains, and the details of the label domains are shown in TABLE VII. We focus on the normal patterns and anomalous patterns when labeling the metrics. If metrics have the same normal and anomalous patterns, we consider that they may belong to the same category. Eventually, the metrics are labeled into 15 categories.

We use accuracy score to evaluate the metric classification performance, where $\text{accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$. We compare the accuracy difference between experimental group and baseline group to verify the effectiveness of metric retrieval model. For experimental group, we use MRG to classify the metrics. For baseline group, we adopt previous classification work to classify the metrics. In addition, it is difficult to evaluate the correctness of the ranking of retrieved metrics. We do not evaluate it here but will discuss it in section VII-B.

2) Experiment results: When designing the baseline models, we focus on evaluating two modules in MRG, namely feature extraction and classification model. We adopt $2 * 2$ factorial design [45] to conduct experiments and evaluate these two factors. For the first factor, the baseline adopts original observations to train classification models, while the experimental group uses extracted features. For the second factor, since MRG introduces GNN layers and Dense layer to classify metrics, we introduce two baseline models to evaluate it. The first one is Random Forest (RF) classifier [46], which is the representative classification method. The second one is only Dense layer, which is used to evaluate the performance of GNN layers.

TABLE VI
STATISTICS OF EVALUATION SAMPLES FOR METRIC RETRIEVAL

Metric Num	Window Size	Label Domain Num	Label Size
22400	2 days	8	15

TABLE VII
THE LABEL DOMAIN FOR METRIC CLASSIFICATION

Metric Domain	Label Domain
Normal Patterns [44]	continuity VS discreteness stationarity VS non-stationarity periodicity VS non-periodicity high variance VS low variance up trend VS down trend VS no trend
Anomalous Patterns [25]	up VS down VS fluctuation sudden VS steady continuous VS transient

Fig 5 presents the performance comparison results between MRG and baselines. For the feature extraction factor, we can observe that the experimental group has a better performance no matter what classification models are adopted. It illustrates that the feature extraction method can extract important information in metrics so that the overall performance is significantly improved. For the classification model factor, we can find that random forest classifier performs better when trained by original observations. When using the extracted features to train model, MSG outperforms baselines and random forest classifier performs worst. The previous result is rooted in the size of model parameters of GNN layers and Dense layer. When using original observations to train the model, the input size is $1440 \text{ per day} \times 2 \text{ day} = 2880$, which may result in large size of model parameters. We need more samples to train the GNN layers and Dense layer when directly using original observations. However, the sample size in the industrial environment is difficult to meet this requirement, which also illustrates the importance of feature extraction. When using extracted features, MSG can leverage and process both the topology information between metrics and time series information within metrics, which results in better performance. In summary, due to the better classification performance, the proposed MSG is effective for recommending unmonitored anomaly-related metric.

VII. DISCUSSION

A. Practical Implication

The practical implication of this paper is primarily rooted in that graph learning is suitable for characterizing the relationship between metrics. This paper introduces graph to

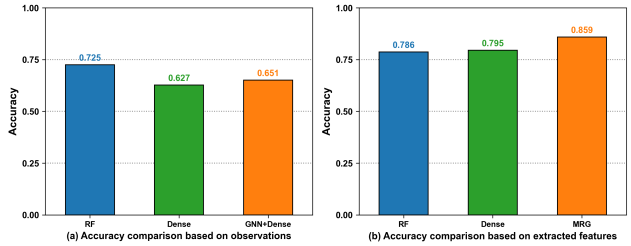

Fig. 5. The metric classification performance comparison results between MRG and baselines.

represent the anomalous relationship between metrics and adopt graph learning to process the topology information of metrics. Comparing with traditional approaches, graph learning can better characterize the unstructured topology of metrics. The experimental results have also shown that graph learning can effectively improve the performance of metric recommendation. In addition, in the industrial environment, metrics are not only used to detect and locate anomalies, but also for further analysis such as load balancing, hotspot discovery and analysis. Due to the capacity of characterizing the relationship between metrics, graph learning can also assist software engineers in further analysis in these new scenarios.

B. Limitations of Metric Recommendation Service

The limitation of the proposed metric recommendation service lies in the ranking of retrieved metrics. As we mentioned in section VI-A, the evaluation of the proposed MSG is designed as the combination of metrics selection and anomaly detection. The ranking of retrieved metrics should be associated with fault analysis and diagnosis for evaluation. However, we have not yet accomplished relevant fault diagnosis methods to automatically find the root cause from anomalous and unmonitored metrics. The unmonitored anomaly-related metrics are recommended to engineers, who will analyze the root cause manually. In future work, we will design an automatic fault diagnosis methods to locate the root cause, and combine with it to improve the ranking of retrieved metrics.

C. Threats to Validity

The internal threat to validity mainly lies in the difference between the proposed methodology and evaluation. The proposed MSG and MRG aim to select metrics for anomaly detection and retrieve metrics for fault diagnosis. But the evaluations of MSG and MRG focus on anomaly detection performance and metric classification performance. To mitigate this threat, we have controlled irrelevant variables and carefully designed the baseline methods.

The external threat to validity mainly lies in the extended application in monitor systems from other companies. We implement the metric recommendation service based on monitoring platform in Tencent. There may exist differences between software architectures, which makes it difficult for other companies or groups to directly implement similar metric recommendation service. To mitigate this threat, we split the service into atomically reproducible components. Our method can be implemented in other companies or groups based on these components.

VIII. Related Work

Anomaly Detection. Current anomaly detection work can mainly be categorized into traditional statistical methods [1], [2] and deep learning based methods [3]–[16]. Deep learning based methods are popular in recent years due to their powerful capacity in modeling complex data. However, no matter which method is applied, the selected metrics matter. The proposed method in this paper can aid prior work in selecting effective metrics for anomaly detection in online systems.

Metric Selection for Anomaly Detection. There also exist some work [34]–[36] that provides automatic mechanism for selecting metrics for anomaly detection. For example, Fu [34] selected metrics that can maximize the mutual information [47] between the selected metrics and the historical faults. Farshchi et.al. [35] leveraged the correlation between metrics and the operator's activity log to select metrics based on regression analysis. Guan et.al. [36] selected the most relevant principal components of different failure types to identify anomalies in cloud infrastructures. However, all of them heavily rely on some external information such as the time and types of historical faults to perform metric selection, rendering these methods difficult to apply in practice.

Fault Diagnosis. A considerable amount of literatures [17]–[25] has investigated fault diagnosis in online systems. For example, MicroCause [24] designs a path condition time series algorithm to learn the relationships between metrics and use random walk to infer the root cause. PatternMiner [25] uses anomaly pattern classification to pick out important anomaly patterns set by engineers and perform root cause metric ranking. Dejavu [17] train a classifier to diagnose recurring failures in online systems. In general, fault diagnosis can be performed from different perspectives, so the exact problem formulations of different work can be quite different. The proposed method in this study can be applied together with prior methods to facilitate fault diagnosis in online systems.

IX. Conclusion

This paper proposes a metric recommendation service for online systems on the basis of graph learning. After an investigation on prior metric selection practice, we design metric recommendation mechanisms for two essential usage scenarios of failure management in online systems, namely metric selection for anomaly detection and metric retrieval for fault diagnosis. Extensive experiments confirm the effectiveness of the proposed approach. Moreover, the proposed approach has been successfully merged into the failure management practice in Tencent® online systems.

X. Data Availability

The KPI dashboards used in our empirical study are in https://anonymous.4open.science/r/submission-9-73C8. Other data used in this paper will be added if we gain the approval of the company. The data will be turned into archived open data in case of acceptance.

REFERENCES

[1] A. Siffer, P. Fouque, A. Termier, and C. Largouët, "Anomaly detection in streams with extreme value theory," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 2017, pp. 1067–1075.

[2] M. Ma, S. Zhang, J. Chen, J. Xu, H. Li, Y. Lin, X. Nie, B. Zhou, Y. Wang, and D. Pei, "Jump-starting multivariate time series anomaly detection for online service systems," in *2021 {USENIX} Annual Technical Conference ({USENIX} {ATC} 21)*, 2021, pp. 413–426.

[3] T. Huang, P. Chen, and R. Li, "A semi-supervised vae based active anomaly detection framework in multivariate time series for online systems," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1797–1806.

[4] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. ACM, 2018, pp. 187–196.

[5] Z. Li, W. Chen, and D. Pei, "Robust and unsupervised KPI anomaly detection based on conditional variational autoencoder," in *37th IEEE International Performance Computing and Communications Conference, IPCCC 2018, Orlando, FL, USA, November 17-19, 2018*. IEEE, 2018, pp. 1–9.

[6] W. Chen, H. Xu, Z. Li, D. Pei, J. Chen, H. Qiao, Y. Feng, and Z. Wang, "Unsupervised anomaly detection for intricate kpis via adversarial training of VAE," in *2019 IEEE Conference on Computer Communications, INFOCOM 2019, Paris, France, April 29 - May 2, 2019*. IEEE, 2019, pp. 1891–1899.

[7] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S. Ng, "MAD-GAN: multivariate anomaly detection for time series data with generative adversarial networks," in *Artificial Neural Networks and Machine Learning - ICANN 2019: Text and Time Series - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part IV*, ser. Lecture Notes in Computer Science, vol. 11730. Springer, 2019, pp. 703–716.

[8] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 1409–1416.

[9] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 2828–2837.

[10] L. Shen, Z. Li, and J. T. Kwok, "Timeseries anomaly detection using temporal hierarchical one-class network," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[11] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "USAD: unsupervised anomaly detection on multivariate time series," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. ACM, 2020, pp. 3395–3404.

[12] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang, "Multivariate time-series anomaly detection via graph attention network," in *20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020*. IEEE, 2020, pp. 841–850.

[13] L. Dai, T. Lin, C. Liu, B. Jiang, Y. Liu, Z. Xu, and Z. Zhang, "SDFVAE: static and dynamic factorized VAE for anomaly detection of multivariate CDN kpis," in *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. ACM / IW3C2, 2021, pp. 3076–3086.

[14] M. Sun, Y. Su, S. Zhang, Y. Cao, Y. Liu, D. Pei, W. Wu, Y. Zhang, X. Liu, and J. Tang, "CTF: anomaly detection in high-dimensional time series with coarse-to-fine model transfer," in *40th IEEE Conference on Computer Communications, INFOCOM 2021, Vancouver, BC, Canada, May 10-13, 2021*. IEEE, 2021, pp. 1–10.

[15] Z. He, P. Chen, X. Li, Y. Wang, G. Yu, C. Chen, X. Li, and Z. Zheng, "A spatiotemporal deep learning approach for unsupervised anomaly detection in cloud systems," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[16] N. Zhao, J. Chen, Z. Yu, H. Wang, J. Li, B. Qiu, H. Xu, W. Zhang, K. Sui, and D. Pei, "Identifying bad software changes via multimodal anomaly detection for online service systems," in *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*. ACM, 2021, pp. 527–539.

[17] Z. Li, N. Zhao, M. Li, X. Lu, L. Wang, D. Chang, X. Nie, L. Cao, W. Zhang, K. Sui, Y. Wang, X. Du, G. Duan, and D. Pei, "Actionable and interpretable fault localization for recurring failures in online service systems," in *Proceedings of the 2022 30th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2022, Nov. 2022.

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

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

[20] J. Lin, P. Chen, and Z. Zheng, "Microscope: Pinpoint performance issues with causal graphs in micro-service environments," in *International Conference on Service-Oriented Computing*. Springer, 2018, pp. 3–20.

[21] P. Chen, Y. Qi, P. Zheng, and D. Hou, "Causeinfer: Automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 1887–1895.

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

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

[24] Y. Meng, S. Zhang, Y. Sun, R. Zhang, Z. Hu, Y. Zhang, C. Jia, Z. Wang, and D. Pei, "Localizing failure root causes in a microservice through causality inference," in *28th IEEE/ACM International Symposium on Quality of Service, IWQoS 2020, Hangzhou, China, June 15-17, 2020*. IEEE, 2020, pp. 1–10.

[25] C. Wu, N. Zhao, L. Wang, X. Yang, S. Li, M. Zhang, X. Jin, X. Wen, X. Nie, W. Zhang, K. Sui, and D. Pei, "Identifying root-cause metrics for incident diagnosis in online service systems," in *32nd IEEE International Symposium on Software Reliability Engineering, ISSRE 2021, Wuhan, China, October 25-28, 2021*. IEEE, 2021, pp. 91–102.

[26] C. Cassagnes, L. Trestioreanu, C. Joly, and R. State, "The rise of ebpf for non-intrusive performance monitoring," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–7.

[27] J. Levin and T. A. Benson, "Viperprobe: Rethinking microservice observability with ebpf," in *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)*. IEEE, 2020, pp. 1–8.

[28] D. N. Jha, G. Lenton, J. Asker, D. Blundell, and D. Wallom, "Holistic runtime performance and security-aware monitoring in public cloud environment," in *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 2022, pp. 1052–1059.

[29] T. Weng, W. Yang, G. Yu, P. Chen, J. Cui, and C. Zhang, "Kmon: An in-kernel transparent monitoring system for microservice systems with ebpf," in *2021 IEEE/ACM International Workshop on Cloud Intelligence (CloudIntelligence)*. IEEE, 2021, pp. 25–30.

[30] I. Baradari, M. Shoar, N. Nezafati, and M. Motadel, "A new approach for kpi ranking and selection in itil processes: Using simultaneous evaluation of criteria and alternatives (seca)," *Journal of Industrial Engineering and Management Studies*, vol. 8, no. 1, pp. 152–179, 2021.

[31] S. Ramadona, S. Haryadi, and D. R. Aryanti, "Over the top call service key performance indicator," in *2015 1st International Conference on Wireless and Telematics (ICWT)*. IEEE, 2015, pp. 1–4.

[32] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, *Site reliability engineering: How Google runs production systems*. " O'Reilly Media, Inc.", 2016.
[33] "Grafana dashboards [acessed at 2022-8-30]," 2022. [Online]. Available: https://grafana.com/grafana/dashboards/
[34] S. Fu, "Performance metric selection for autonomic anomaly detection on cloud computing systems," in *2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*. IEEE, 2011, pp. 1–5.
[35] M. Farshchi, J.-G. Schneider, I. Weber, and J. Grundy, "Metric selection and anomaly detection for cloud operations using log and metric correlation analysis," *Journal of Systems and Software*, vol. 137, pp. 531–549, 2018.
[36] Q. Guan and S. Fu, "Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures," in *2013 IEEE 32nd International Symposium on Reliable Distributed Systems*. IEEE, 2013, pp. 205–214.
[37] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International journal of data mining & knowledge management process*, vol. 5, no. 2, p. 1, 2015.
[38] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.
[39] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," in *Sixth international conference on data mining (ICDM'06)*. IEEE, 2006, pp. 613–622.
[40] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018.
[41] M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. Bota, H. Liu, T. Schultz, and H. Gamboa, "Tsfel: Time series feature extraction library," *SoftwareX*, vol. 11, p. 100456, 2020.
[42] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.
[43] J. Turnbull, *Monitoring with Prometheus*. Turnbull Press, 2018.
[44] T.-c. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, 2011.
[45] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.
[46] A. Paul, D. P. Mukherjee, P. Das, A. Gangopadhyay, A. R. Chintha, and S. Kundu, "Improved random forest for classification," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 4012–4024, 2018.
[47] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical review E*, vol. 69, no. 6, p. 066138, 2004.