IEICE TRANS. INF. & SYST., VOL.E106–D, NO.5 MAY 2023

PAPER Special Section on Data Engineering and Information Management

MicroState: An Anomaly Localization Method in Heterogeneous Microservice Systems

Jingjing YANG[†], ***Nonmember***, **Yuchun GUO**[†a)], ***Member***, ***and*** **Yishuai CHEN**[†], ***Nonmember***

SUMMARY Microservice architecture has been widely adopted for large-scale applications because of its benefits of scalability, flexibility, and reliability. However, microservice architecture also proposes new challenges in diagnosing root causes of performance degradation. Existing methods rely on labeled data and suffer a high computation burden. This paper proposes MicroState, an unsupervised and lightweight method to pinpoint the root cause with detailed descriptions. We decompose root cause diagnosis into element location and detailed reason identification. To mitigate the impact of element heterogeneity and dynamic invocations, MicroState generates elements' invoked states, quantifies elements' abnormality by warping-based state comparison, and infers the anomalous group. MicroState locates the root cause element with the consideration of anomaly frequency and persistency. To locate the anomalous metric from diverse metrics, MicroState extracts metrics' trend features and evaluates metrics' abnormality based on their trend feature variation, which reduces the reliance on anomaly detectors. Our experimental evaluation based on public data of the Artificial intelligence for IT Operations Challenge (AIOps Challenge 2020) shows that MicroState locates root cause elements with 87% precision and diagnoses anomaly reasons accurately.

key words: *root cause analysis, element invocation, microservice system*

1. Introduction

With the advantage of flexible scalability and fast delivery, microservice architecture (MSA) has been widely adopted in the modern IT industry, such as the Internet of Things [1], mobile computing, and cloud computing [2]. With MSA, an application is decomposed into single-function and loosely coupled microservices that intercommunicate with lightweight protocols [3]. Each microservice runs as a set of instances that can be deployed on multiple distributed machines. Hence, MSA-based systems contain numerous elements and processes, with complex system structures and dynamic microservice interactions, where detecting and diagnosing system performance is particularly important.

When the system anomaly occurs, operators must troubleshoot it and locate the first microservice where the anomaly originates (i.e., the root cause). We present a toy example of a microservice system with anomalies in Fig. 1, an anomaly which originates from S_1 propagates to its dependent microservices S_2 and S_3, the system performance will continuously degrade until S_1 is fixed. Based on invocation records between microservices (tracing data) and resource consumptions of elements (metric data), operators can track the execution processes of the user request and related resource information, to infer the root cause element and detailed reasons of the anomaly, e.g., S_1 failed by CPU hog.

To pinpoint the root causes, various root cause localization methods have been proposed. Machine learning-based methods [4], [5] identify the root cause based on the labeled data which is difficult to fetch from frequently updated microservice systems. Graph-based methods [6]–[8] construct the microservice dependency graph and infer the root causes based on the walking algorithm. Since the microservice system is highly dynamic, such a dependency graph needs to be updated frequently. Under the huge monitoring data generated by the microservice system, machine learning-based methods and graph-based methods suffer the computation burden. Hence, an unsupervised and lightweight localization method is required.

There are four challenges to achieving this goal. *1) Huge amount of data.* Microservice system contains numerous elements which will generate a large amount of monitoring data continuously, as long as the system runs. Such a huge amount of data hides valuable clues to pinpoint the culprit, posing a daunting challenge for algorithm design of timely anomaly diagnosis. *2) Dynamic invocations.* Due to multiple reasons, such as dynamic system workload, load-balanced mechanism, and system design patterns, microservices interact flexibly with each other. It is difficult to find a stable pattern of microservice's normal invocations to detect anomalies. *3) Element heterogeneity.* To provide enough resources for dependently deployed microservices, MSA adopts resource virtualization techniques. Hence, MSA contains heterogeneous elements such as physical machines, virtual operation systems, and containers. These elements undertake diverse functions developed with

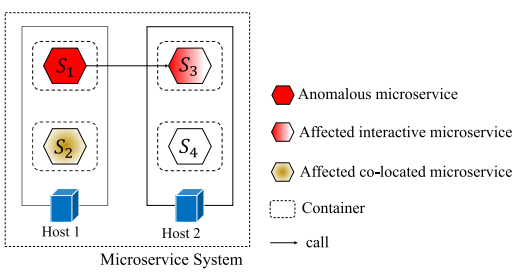


Fig. 1 Example of microservice system with anomalies.

Manuscript received June 19, 2022.
Manuscript revised November 15, 2022.
Manuscript publicized January 13, 2023.

[†]The authors are with the Beijing Jiaotong University, Beijing, 100044, China.
a) E-mail: ychguo@bjtu.edu.cn (Corresponding author)

DOI: 10.1587/transinf.2022DAP0003

polyglot design, and have different characteristics of invocation latency. *4) Various metrics.* Microservice system monitors various metrics with diverse characteristics, such as the value range, periodicity, collection frequency, and anomaly symptoms. Analyzing and detecting these metrics properly is challenging because deterministic threshold schemes lack effectiveness and customized methods are burdened.

To resolve the above issues, we propose a novel root cause localization method, which is lightweight and unsupervised, named MicroState. First, MicroState pinpoints the culprit element, and then, it identifies the detailed reason for the culprit's abnormality, i.e., the root cause metric. In order not to rely on labeled data, MicroState converts the quantification of object anomalies into the assessment of the object's self-state variation.

For root cause element location, firstly, MicroState characterizes the element state from its invocations by generating the element's invocation feature vector, which processes a huge amount of tracing data into small-scale vectors and mitigates the computation burden. Secondly, to mitigate the state fluctuation caused by dynamic invocation, MicroState adopts the DTW algorithm to quantify an element's abnormality, which finds the maximum similarity of the element's states at two-time slots. Thirdly, MicroState groups similar elements and infers the anomalous group, to avoid the interference of different abnormality symptoms from heterogeneous elements. Finally, MicroState locates the root cause element by filtering false alarms based on the anomaly frequency and persistency. For root cause metric identification, MicroState characterizes the metric state by extracting its trend features, and processes the continuous values into finite trends. MicroState infers the root cause metric based on its state variation, to avoid customized anomaly rules for different metrics.

MicroState is lightweight since it does not identify the root cause from a huge amount of data, but characterizes the object state, i.e., element state and metric state. MicroState is unsupervised since it requires no labeled data, but quantifies an object's abnormality through its state variation. We evaluate MicroState using a dataset collected from a large wireless provider. Experimental results show that MicroState has a high diagnostic precision compared to baselines. In summary, our contributions include:

- For anomaly detection, we characterize the state of the detected object (i.e., element and metric), to mitigate the computation burden from a huge amount of data; we convert the quantification of objects' abnormality into the assessment of objects' state variation, which requires no labeled data. The detailed processes are presented in the modules of feature vector generation and the root cause metric location.
- For root cause element location, we adopt the DTW algorithm to calculate the variation of element state, which mitigates the interference from dynamic invocations, presented in the anomaly assessment module; we group similar elements and define an index of element group to infer anomalous group, which addresses the element heterogeneity challenge, presented at the module of anomaly group inference.
- We design a lightweight and unsupervised algorithm, MicroState, to pinpoint the root cause element with detailed reasons. We conduct experiments on an open dataset system including baseline comparison and ablation study. Experimental results demonstrate the effectiveness of MicroState.

The rest of this paper is organized as follows: we summarize the related work in Sect. 2. Section 3 defines the problem and presents the overall framework and details of MicroState, including the (invoked) state-based root cause element location and the (trend) state-based metric location. We evaluate and discuss MicroState in Sect. 4 and conclude the paper in Sect. 5.

2. Related Work

Various research efforts have been devoted to diagnosing performance issues in microservices. Overall, these approaches employ machine learning, graph analysis, and causal inference to infer the root cause. The observation data can be divided into system logs, tracing data, and metrics. We briefly introduce related work from two aspects: service granularity location and metric granularity location.

Service granularity location. Researchers have proposed different approaches to pinpoint root cause services, including graph-based approaches, machine learning-based approaches, and time series-based methods [9].

Graph-based approaches [6], [7] generated a topology graph from microservices execution paths at each period. Based on such graphs, MonitorRank [6] determined root causes by running the personalized PageRank algorithm. MicroHECL [7] expanded the anomaly propagation path. When anomaly propagation chains cannot be further extended, the services at the end of each determined propagation chain are root causes. Meng et al. [10] assessed the anomaly degree of traces with tree edit distance. CloudDiag [11] processed traces to determine "anomalous groups" considering service call tree and response times variance.

Machine learning-based approaches were proposed to diagnose anomalies [12]–[14]. Sage [13] determined the root cause of unpredictable performance through deep generative models and leveraged unsupervised models. Seer [12] diagnosed spatial and temporal patterns that translated to QoS violations with the deep learning methods. TraceAnomaly [14] detected performance anomalies utilizing a deep Bayesian neural network. The last invoked services in anomalous service interaction sequences were root causes.

However, both graph-based methods and machine learning-based methods suffer a huge computation burden and are difficult to deploy on large systems online. MicroState generates an element's state as a vector based on its

invocation features and extracts the metric's state based on its trend feature. The location method is based on the small-scale vector calculation which is lightweight.

Metric granularity location. A series of research use causal inference to infer the root cause metric. Different causal inference methods have been applied to learn the causal relations among metrics. The Sieve [15] and Loud [8] systems construct anomaly propagation graphs across all metrics using the Granger causality test. CauseInfer [16] infers the root cause by constructing a metrics causality graph for each service with PC algorithm [17], services are connected by a dependency graph. MicroCause [18] captured the sequential relationship of time series data and a novel temporal cause-oriented random walk method that integrates the causal relationship, temporal order, and priority information of monitoring data. MicroDiag [19] constructs a component dependency graph and infers the causal relations among metrics with different causal inference techniques. It derives a metrics causality graph and infers root causes by the PageRank algorithm.

Existing studies only consider few metrics of the microservices in the diagnosing process, however, a microservice may be monitored with a large number of indicators comprehensively. Casual inference-based methods will generate a large number of spurious causal-effect relationships under a large number of tested variables. In MicroState, we consider diverse types of metrics. We locate the root cause metrics of the targeted element among tens of metrics based on the trend features extracted from the initial metric data.

3. Method

Firstly, we give the problem statement of root cause localization in this paper. We define containers, virtual systems, and physical hosts as the elements of microservice systems. Our data consists of elements' tracing records and metric data from n elements. Suppose during an incident period d, system performance degrades. Given a set of elements, their invocation records, and their metric data, our goal is to identify 1) the culprit element e_{rc} that initiates the anomalies and 2) the detailed reason m_{rc} of the anomaly. The output of our algorithm is top-k root cause elements with detailed reasons. The optimization goal is the precision of top-k elements defined at Sect. 4.1.

Tracing data contain execution paths and time-consuming records, which are used to pinpoint the root cause element. Metric data record the element's resource consumption, such as CPU or memory utilization, which can be used to diagnose detailed reasons for anomalies. An example of a user request and related trace records is presented in Fig. 2. As shown in Fig. 2, the trace records of a unique request can be modeled as a directed weighted graph, where the nodes indicate services and the weights of edges indicate the time consumed by the service responses.

Figure 3 shows the overview of MicroState. Once system anomalies are detected, MicroState pinpoints the root cause element based on the aggregated tracing data, and locates the detailed anomaly reason based on the metric data. Firstly, MicroState characterizes the invocation states, i.e., the feature vectors of elements, and quantifies their abnormality through a warp-based state comparison. MicroState groups similar elements together and infer the anomalous group. And root cause element is located by a rank procedure, where ranks more frequent and longer anomalous elements with higher priority, to filter false alarms. Then, to find the root cause metric, MicroState extracts the metric's trend features, which describe the metric's trend pattern, to find the real anomaly reason among diverse metrics with different symptoms.

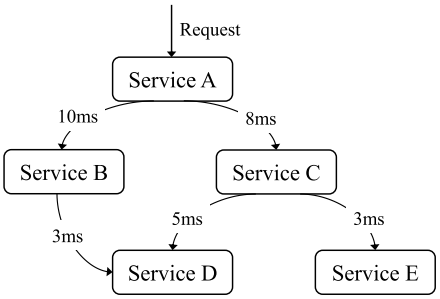


Fig. 2 Request example. There is a call-to-call relationship between the microservices, the time consumed by the service responses is shown in the directed edges between services.

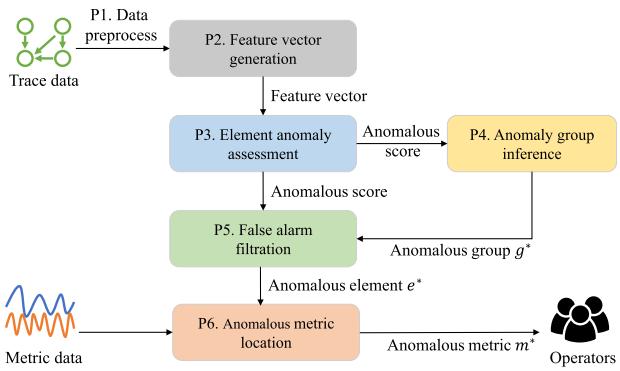


Fig. 3 MicroState framework.

3.1 Root Cause Element Location

MicroState locates the root cause element based on the invocation state of elements. It pinpoints the culprit through three procedures: quantifying elements' abnormality, inferring the anomalous group, and filtering false alarms. Firstly, MicroState characterizes elements' invocation state, i.e., feature vector, considering all its invocation relationships. Secondly, MicroState quantifies elements' abnormality based on their self-state variance at two-time slots through a warped procedure, to mitigate the interference of dynamic invocations. Thirdly, MicroState groups heterogeneous elements with similar invocation characteristics to infer the group which contains the root cause element, to avoid inaccurate results by directly inferring the root cause

Table 1 Symbolic description.

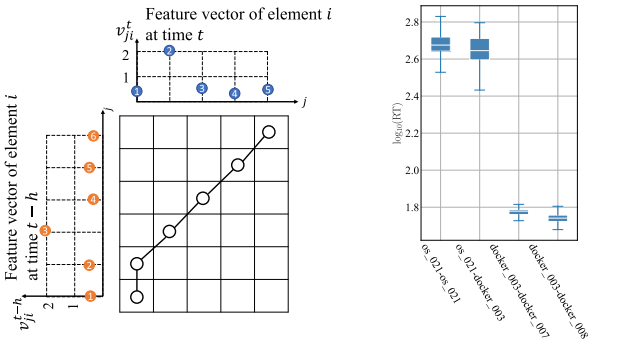
Notations	Definitions
n, c	Number of elements; types of elements
d	Interval of anomaly
e_{rc}, m_{rc}	Set of root cause elements; set of root cause metrics
V_i^t, v_{ji}^t	Invoked time vector of element i at timestamp t; invocation time among element j and invoked element i at timestamp t
E_i^t	Anomalous score of element i at timestamp t
G_j^t	Anomalous score of element group j at timestamp t
R_x	Ranking score of candidate element x
$m_{i,k}^t$	$No.k$ metric of element i at timestamp t
$T_{i,k}^t$	Trend label of $No.k$ metric of element i at timestamp t
$S_{i,k}^t$	Ranking score of candidate metric k of element i at timestamp t
g^*, e^*, m^*	Anomalous group; Anomalous element; Anomalous metric
α, β, γ	Threshold parameters
h, h_1, h_2	Time intervals of state comparison

element. Finally, MicroState locates the root cause element from the anomalous group with a ranking procedure to filter false alarms.

1) Feature vector generation. Diagnosing anomalies based on the time series of invocation pairs will cause false alarms and missing alarms, MicroState characterizes the element's complete state by generating its invocation feature vector with considering all its invocation relationships. Since directly analyzing the huge amount of tracing data is burdened, we aggregate trace data at minute-level granularity. Assumes there are n elements of c types in the system. The element type is divided from the element's basic functions. For example, physical machines, virtual machines, docker, and databases are different element types, docker is a dependent microservice instance, and system data is stored in databases.

The execution paths are split into invocation pairs, i.e., the caller element and the callee element. Then, we compute the average latency of each invocation pair for one minute. At each time slot t, the invocation vector of the element i is represented as $V_i^t = [v_{1i}^t, \ldots, v_{ji}^t, \ldots], j \in 1, 2, \ldots, n$, v_{ji}^t indicated the average latency of the caller element j and the callee element i. The length of V_i^t is the number of all caller elements of the callee element i at time t, and the index of the invocation element i is arranged by the element type.

2) Element anomaly assessment. Owing to the challenge of flexible and variable invocations, it is difficult to distinguish true anomalies and variable but normal invocations. Based on the elements' feature vectors, MicroState quantifies their abnormality by a self-state comparison. Inspired by the application of Dynamic Time Warping (DTW) [20] algorithm in time series, we use the DTW algorithm here to handle the interference of variable invocations. We define the element's anomaly score as the DTW distance between its feature vectors at different time slots. Since the DTW algorithm warps two series (replaced as vectors in this paper) as similar as possible to find an optimal match, such a warping procedure finds the maximum similarity of two vectors. Hence, it can mitigate the variation of an element's



(a) DTW of feature vectors

(b) Response time (RT) of heterogeneous elements

Fig. 4 Schematic diagram of MicroState modules. (a) shows a warping example of the feature vectors. Colored nodes represent the invocation vectors of the caller element j and the callee element i. The warping path explicitly states which datapoints of vector v_i^t align with what datapoints of vector $v_i^{(t-h)}$, such warping procedure finds the optimal match of two vectors. (b) presents an example of response time (RT) between different elements. The horizontal axis is the four selected call pairs and the vertical axis is the logarithmic scale of the related response time (RT).

state affected by variable invocations. A warping example of the feature vectors is shown in Fig. 4 (a). Two feature vectors of element i at time t ($t-h$) are warped. We define E_i^t as the anomalous degree of the element i at time t

$$E_i^t = \widetilde{d}(V_i^t, V_i^{t-h}) \tag{1}$$

V_i^{t-h} is the invocation vector at $t-h$. $\widetilde{d}(V_i^t, V_i^{t-h})$ is the DTW distance of the invocation vector V_i^t and the invocation vector V_i^{t-h}.

To the best of our knowledge, we are the first to use the DTW algorithm to warp two feature vectors, not the time series. We use the DTW algorithm here for three reasons. 1) DTW algorithm can warp different length vectors, which can fit the frequently updated microservice systems. 2) DTW algorithm computes the Euclidian distance which is reasonably used to measure the latency variation of the target element. 3) Small DTW values indicate element state does not change dramatically, which denotes there are no anomalies since h is much larger than the anomaly period.

3) Anomaly group inference. Due to the element heterogeneity, directly locating the root cause based on the anomalous scores is inefficient. An example of response time (RT) between different types of elements is presented in Fig. 4 (b). We took two hours without anomalies and chose the logarithmic scale for clearer visualization. It can be observed that invocations of os-type elements exhibit greater volatility compared to docker-type elements.

Based on the observation that the invocation characteristics of the same type elements are similar, but there are big differences between the different type elements, hence, we group the elements of the same type together. Since the purpose of grouping is to compare elements with similar invocation characteristics together; other clustering methods based on the element's latency characteristics (e.g., mean value, variance) also can be used in different microservice

systems. It is reasonable that the anomaly degree of the root causes will be significantly different from other normal elements in the group because elements in a group are more likely to present similar invocation features with similar business functions. Hence, we compute the anomalous score of a group based on the comparison of elements' anomalous scores in this group.

We propose an index G_j^t to assess the anomalous degree of the group j at time t. With a given group j, we define m_j as the element that has the maximum anomalous score in the group j. G_j^t is defined as the ratio of $E_{m_j}^t$ to the average anomalous score of the remaining elements.

$$G_j^t = \frac{E_{m_j}^t}{\overline{E_j^t}}, \quad j \in 1, 2, \cdots, c \tag{2}$$

$$\overline{E_j^t} = \frac{\sum_{i \in j, i \neq m_j} E_j^t}{|j| - 1}, \quad j \in 1, 2, \cdots, c \tag{3}$$

We do not compare the anomalous scores of heterogeneous elements directly, but assess the group anomaly degree based on anomalous scores of elements of the same type. In addition to the invocation latency, tracing data also record the success rate of invocations, we utilize the success rate of the group as the average success rate of all elements in the group. If the group has the maximum group anomalous score and the minimum success rate, it can be inferred as the anomalous group g^*; otherwise, g^* is referred to as the group which has the maximum group anomalous score.

4) False alarm filtration. Since some anomalies' symptoms may not last for a notable time. Under the interference of false alarms, it is challenging to locate the true anomalous elements. MicroState gives higher priority to candidates which have longer anomaly duration and higher anomaly frequency, to filter false alarms.

Based on the anomalous group g^* and the anomaly scores of the elements in g^*, the candidate root cause e^* can be inferred as the element that has the highest anomaly score. During the anomaly duration time d, assuming there is one candidate. Hence, the candidate root cause list is $r^d = \{r^0, r^1, \cdots, r^{d-1}\}$. For a candidate, continuously appearing in r^d is a strong signal that it is anomalous. MicroState utilizes such anomaly persistency to locate the root cause. For candidate x in r^d, it may appear multiple times or at consecutive slots. The duration l that x appears continuously ranges from 0 to d. We record each anomaly duration l of candidate x, named as D_x. Assume that $r^d = [A, B, A, A, C]$, then $D_A = [1, 2]$, $D_B = [1]$, and $D_C = [1]$. Then, we compute the final ranking score R_x of candidate x

$$R_x = \sum_{l \in D_x} l^\alpha, \alpha > 1 \tag{4}$$

R_x indicates the probability of candidate x to be a root cause, the longer and more frequently that the element is anomalous, the more likely x to be the root cause. α is the weighted factor to quantify the importance of the duration of x that appears continuously in r^d. We choose a power function here to calculate the final ranking score, and the exponent of the power function α should be set greater than 1, because the longer the duration, the higher the score, α is set as 1.1 in this paper. Other functions can also be used according to actual situations. For example, if a normal element is impossible to appear continuously in the candidate set, α can be set bigger to filter normal elements. The key to filtering false alarms is to take into account the anomaly duration of the candidate.

3.2 Root Cause Metric Location

The performance issues of the microservices need to be addressed timely, it is vital to provide root causes of metric granularity for engineers to take accurate operations further. Since various monitoring metrics have different characteristics, one anomaly detector does not suit all metrics, but it is impossible to customize the anomaly detector for all metrics. Owing to the metric correlation, multiple anomalous metrics appear at the same time. And almost all the existing methods locate the root cause metrics among few metrics, hence, diverse various features of metrics have not gained enough attention.

Since the root cause metric trend will exhibit differently compared to its normal states, such as a sudden increase or a sudden decrease, MicroState processes initial metric data, i.e., time series, into discrete trend features, and locates the root cause based on the metric state variation. Firstly, MicroState defines finite trends of time series: stable, increase, decrease, and other trend. Then, MicroState characterizes the metric's state by extracting the metric's trend from the initial time series, which converts the time series to discrete trends. Finally, MicroState assesses the metric's abnormality through its state variation and pinpoints the metric which has the maximum variation as the root cause.

We define four types of metric trends as Table 2 shows. Given a short time slice of a metric $[m_{i,k}^{t-d+1}, \ldots, m_{i,k}^t]$. The trend of $m_{i,k}^t$ is denoted by a time sliding window with length $d-1$. If the standard deviation of the metric data is 0, then the trend is stable; if $m_{i,k}^t$ is greater (smaller) than the average value in the time window $[t-d+1, t-1]$ multiply the adjustable parameter β (γ), then the trend of $m_{i,k}^t$ is increase (decrease); otherwise, the $m_{i,k}^t$ is in other trend. Referring to typical anomaly patterns summarized by [21], MicroState can detect these anomaly patterns, since anomaly patterns can also be recognized by the combination of different trends. For example, a sudden increase anomaly pattern

Table 2 Trend description.

Label	Trend	Conditions
0	Stable	$\sqrt{\frac{\sum_{t-d+1}^{t}(m^i-\overline{m})}{d}} = 0$
1	Increase	$m^t > \beta(\frac{\sum_{t-d+1}^{t} m^i}{d}), \beta > 1$
2	Decrease	$m^t < \gamma(\frac{\sum_{t-d+1}^{t} m^i}{d}), \gamma < 1$
3	other	otherwise

can be decomposed to the trend sequence as: stable and increase.

During the anomaly duration d, $T^t_{i,k}$ which indicates the trend of $No.k$ metric of element i at timestamp t, we define a comparison function f to infer whether $m^t_{i,k}$ exhibits differently compared to its former state at two different time slot $t-h_1$ and $t-h_2$. h_1 and h_2 should be set much longer than d. The final ranking score $S^t_{i,k}$ of the metric $m_{i,k}$ in the candidate set is determined by the trend comparison results between the detected time t, $t-h_1$ and $t-h_2$. Then, the root cause metrics are located by the final ranking score $S^t_{i,k}$.

$$S^t_{i,k} = \sum_{t-d+1}^{t} f(T^t_{i,k}, T^{t-h_1}_{i,k}, T^{t-h_2}_{i,k}) \tag{5}$$

$$f(x_1, x_2, x_3) = \begin{cases} 0, & if\, x_1 = x_2\ or\ x_1 = x_3 \\ 1, & otherwise \end{cases} \tag{6}$$

For example, if the trend label list of $No.k$ metric of element i in the time window $[t-d+1, t]$ is [0, 1, 1, 2, 0], its trend list at time window $[h_1 - d + 1, h_1]$ is [0, 0, 0, 0, 0], and its trend list at time window $[h_2 - d + 1, h_2]$ is [0, 0, 0, 0, 0]. Then, the $S^t_{i,k}$ is 3. A higher score denotes a bigger possibility of the metric being a root cause, which exhibits a more different trend compared to its normal states. The most likely anomalous metric is referred to as m^* whose anomalous score is the maximum of all metrics. MicroState uses a simple but effective method to extract the metric's trend, and other methods such as lightweight machine learning classifiers can also be used to divide the trend in other datasets.

4. Evaluation

To demonstrate the effectiveness of our algorithm clearly, we present and discuss the result of locating the root cause element in Sect. 4.1 and the root cause metric identification in Sect. 4.2. The overhead, sensitivity, and limitations of MicroState are discussed in Sect. 4.3.

4.1 Root Cause Element Location

Dataset Description and settings. We evaluate MicroState on the open dataset from International AIOps Challenge [22], which is a competition focusing on the diagnosis of microservices systems. Here we use the 2020 version which was collected from a large wireless provider. The average amount of daily tracing data is about 1.2 GB. The dataset provides a complete description of failure types, which are mainly divided into three categories: container-docker (docker), virtual host (os), and database (db). The system architecture contains 22 virtual hosts, 13 databases, and 8 dockers. The counts of fault types are shown in Table 3. h is set as 60 in experiments.

Evaluation Metric. To quantify the performance locating algorithm, we use the performance metric: precision at top k. It denotes the probability that the top k results given by an algorithm include the real root cause, denoted as

Table 3 Count of failure type.

Element type	Fault number
container-docker	49
os	17
database	12
all	78

Table 4 Performance of element localization.

Element	MicroState	Cauchy	MicroRCA	RS
Overall				
PR@1	0.55	0.15	0.05	0.06
PR@3	0.71	0.29	0.21	0.15
PR@5	0.87	0.53	0.46	0.23
Docker				
PR@1	0.47	0.22	0	0.05
PR@3	0.61	0.39	0.18	0.13
PR@5	0.82	0.63	0.55	0.21
Os				
PR@1	0.47	0.06	0.24	0.06
PR@3	0.76	0.12	0.35	0.18
PR@5	0.94	0.24	0.35	0.27
Database				
PR@1	1	0	0	0.06
PR@3	1	0.17	0.25	0.18
PR@5	1	0.5	0.25	0.27

PR@k. A higher *PR@k* score, especially for small values of k, represents the system correctly identifies the root cause, Let $R[i]$ be the rank of each cause and e_{rc} be the set of root causes. More formally, *PR@k* is defined on a set of given anomalies A as:

$$PR@k = \frac{1}{|A|} \sum_{a \in A} \frac{\sum_{i<k} (R[i] \in e_{rc})}{(\min(k, |e_{rc}|))} \tag{7}$$

Baseline Methods. We compare our algorithm to three baseline methods as follows.

- *Cauchy*: The champion method of AIOps Challenge 2020 [23] uses an anomaly detector (Cauchy detector) to detect the response time of each call pair, takes the call pair's anomaly score as the invoked element's anomaly score, and calculates the average anomaly score for all call pairs of an element, then the root cause is located based on the ranking of element scores.
- *MicroRCA*: MicroRCA [24] constructs attributed graph model of microservices and correlates service anomalous performance symptoms with corresponding resource utilization. It adopts a personalized PageRank to infer root causes.
- *Random Selection (RS)*: Random selection is a way engineers use when lacking domain-specific knowledge of the system. Every time, they randomly select an unchecked microservice to investigate until the root cause is found.

Evaluation and comparison. Table 4 shows the performance of MicroState and baselines in identifying the root causes of three types of anomalies. Overall, MicroState achieves an average *PR@5* in the range of 80%–100%

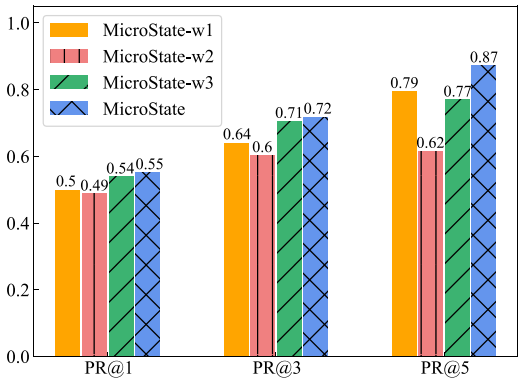


Fig. 5 Performance of ablation experiment in terms of PR@1, PR@3 and PR@5.

among all anomaly types. We evaluate the performance of MicroState by comparing it with three baseline methods, i.e., Cauchy, MicroRCA, and RS. We apply MicroState and three baseline methods to all anomaly cases and get their performance in terms of *PR*@1, *PR*@3, and *PR*@5. We can see that all these methods can not pinpoint the culprit metric in the top 1 of the ranked list. However, compared to the best baseline method, our MicroState achieves an improvement of 64% in precision, and it achieves 71% in*PR*@3 and 87% in *PR*@5.

Results of MicroRCA show inefficiencies in locating os-type and db-type anomalies, because 1) MicroRCA does not consider these types of anomalies, and 2) the interference of element heterogeneity. MicroRCA locates root causes based on trace data and metric data while MicroState only needs trace data. Besides, we can see MicroState's precision in identifying docker-type anomalies is lower than other type anomalies, because docker-type anomalies involve several different kinds of anomalies, such as CPU fault anomaly, network delay anomaly, and network loss anomaly. Since the network-related anomalies will not cause latency deviation of the execution paths, such anomalies may not express in tracing data and are difficult to locate.

Ablation study. We assess the effectiveness of MicroState modules by applying ablation experiments on the root cause element location.

- *MicroState-w1*: MicroState quantifies microservice abnormality based on the euclidean distance of two feature vectors, instead of the DTW algorithm.
- *MicroState-w2*: MicroState infers the root cause element directly based on the anomaly score, without the anomalous group inference.
- *MicroState-w3*: MicroState locates the root cause element based on the frequency of candidate elements, without the false alarm filtration.

Figure 5 presents the performance of MicroState and ablation experiments, in terms of all cases at PR@1, PR@3, and PR@5. We can see that all modules improve the efficiency of MicroState. In terms of PR@3 and PR@5, anomaly group inference plays a vital role. MicroState-w1 only assesses the effectiveness of the DTW algorithm, not the whole anomaly assessment module, whose effectiveness can be approximately evaluated by the comparison with the baseline method: Cauchy.

Table 5 Performance of metric localization.

Element	Cauchy	M-Cauchy	FFT	M-FFT
Overall				
PR@1	0.31	0.62	0	0.10
PR@2	0.69	0.83	0.14	0.41
PR@3	0.83	0.90	0.21	0.72
Os				
PR@1	0.47	0.65	0	0.12
PR@2	0.65	0.76	0	0.29
PR@3	0.71	0.88	0	0.82
Database				
PR@1	0.08	0.58	0	0.08
PR@2	0.75	0.92	0.33	0.58
PR@3	1.0	0.92	0.50	0.58

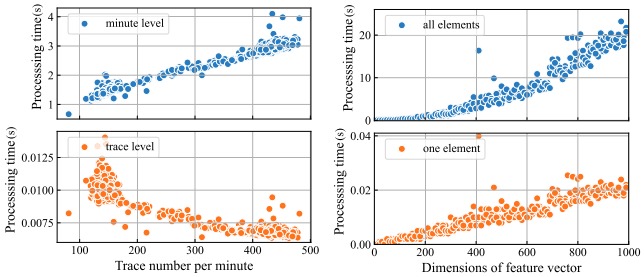
4.2 Root Cause Metric Location

Datasets. The dataset is the metric data collected from the same microservices system described in Sect. 4.1. The dataset contains several metrics of different elements. The metric number of the os-type element is 52. The metric number of the database-type element is 47. Since the metric amount of docker-type elements is less than 10 and is easy to be located, we only focus on the os-type and database-type anomalies in this subsection. Trend parameters β is set as 1.2, γ is set as 0.6. h_1 is set as 60 and h_2 is set as 120 in experiments.

Benchmarks. To illustrate the effectiveness of the trend comparison, we choose two anomaly detectors as baselines. M-Cauchy (M-FFT) denotes the Cauchy (FFT) detector deployed with the trend comparison of MicroState.

- *Cauchy*: An anomaly detector used in [25] to detect the time series abnormalities, which assumes data has the Cauchy distribution.
- *FFT*: An anomaly detector proposed by [26] based on the spectral residual.

We apply two baseline methods to database and os anomaly cases, and get the average performance of each method in terms of PR@1, PR@2, PR@3, as shown in Table 5. We can see that compared to the baseline methods, our MicroState achieves a large improvement with at least 76.5% in precision, and it achieves 97% in PR@2 and 100% in PR@3. In os-type anomalies, we can see MicroState achieves a higher performance in the FFT detector than the Cauchy detector, because there are many anomalous metrics in os-type anomaly cases, which cause strong interference to locate the culprit. MicroState mitigates such interference and finds the true root cause metrics. In db-type anomalies, we can see in most cases, MicroState improves the location precision, but in the PR@3 case, Cauchy outperforms MicroState, the reason is that some anomalies present



(a) Processing time of feature vector generation (b) Processing time of DTW

Fig. 6 Processing time of MicroState modules.

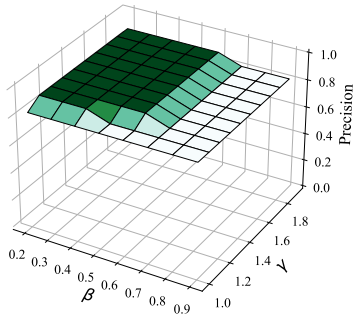


Fig. 7 Performance against trend parameters β and γ.

significant bursting characteristics which are easier caught by the Cauchy detector, but MicroState misses the true root cause under the interference of other metrics.

4.3 Discussion

Here, we discuss the overhead and the sensitivity of MicroState.

1) Overhead: The overhead of MicroState is mostly caused by two modules: feature vector generation and element anomaly assessment which quantifies the elements' abnormality through the DTW algorithm. Figure 6 shows the processing time of these two modules, the vertical axis is in seconds. We generate feature vectors in one minute, and there are different trace numbers in one minute. Figure 6 (a) presents the processing time of feature vector generation against trace numbers per minute. As the blue dots show, when the trace numbers increase, the processing time increases, but for each trace's processing time, as the orange dots show, it decreases. Figure 6 (b) shows the processing time of the DTW algorithm against different dimensions of feature vectors. Since the vector dimension in our dataset is less than 100. We simulate vectors with different dimensions, ranging from 0 to 1000. We can see that the processing time of the DTW algorithm is short enough to run in a real-time scenario. The blue dots present the processing time of all elements and the orange dots show the processing time for one element. With the element number increases, the processing time of one element increases approximately linearly, even if the system is huge, it won't take much computing time.

2) Sensitivity: To evaluate the sensitivity of MicroState to the trend feature parameters, β and γ. We analyze the performance of MicroState with different values of them, in terms of os-type failures. Figure 7 shows the precision of MicroState in terms of PR@3, where β ranges from 0.2 to 1, γ ranges from 1 to 2. We can see the performance of MicroState almost changes slightly with the parameters changing, because the core idea of root cause metric location is to convert continuous time series values into discrete trends, with a time-sliding window. Hence, the performance of MicroState is not sensitive to parameters β and γ. In practice, engineers can tune the parameters according to the metric data features.

3) Limitations: MicroState classifies elements into 3 types: virtual host, docker, and database. Such classification may not suit other microservice systems. Based on elements' invocation characteristics, clustering methods can be used to group similar elements. In the module of false alarm filtration, MicroState only chooses one function, i.e., the power function, to calculate the ranking score of the candidate, different ranking methods based on element anomaly duration and frequency will be explored further.

5. Conclusion

In this paper, we proposed a new method, i.e., MicroState, to diagnose root causes with detailed reasons in heterogeneous microservices systems. MicroState converts the abnormality quantification into the state variation of the detected object. For the detected element, MicroState characterizes its invocation state; for the detected metric, MicroState extracts its trend features. MicroState quantifies the root cause element based on its state variation, then locates the anomalous group first, and pinpoints the culprit element with the filtration of false alarms. MicroState pinpoints anomaly reasons at a fine granularity on the metric level and finds the root cause metric based on the metric state variation. Our experimental results showed that MicroState achieves 87% in the precision of the location of root cause elements and performs better than anomaly detectors which directly detect metric data.

Acknowledgments

Thanks to the public dataset provided by TSINGHUA NETMAN LAB. This work was supported by the National Natural Science Foundation of China under Grant No. 61872031.

References

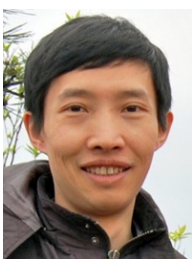
[1] B. Butzin, F. Golatowski, and D. Timmermann, "Microservices approach for the internet of things," 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), pp.1–6, IEEE, 2016.

[2] P.D. Francesco, I. Malavolta, and P. Lago, "Research on architecting microservices: Trends, focus, and potential for industrial adoption," 2017 IEEE International Conference on Software Architecture (ICSA), pp.21–30, IEEE, 2017.

[3] S. Newman, Building microservices, O'Reilly Media, 2021.
[4] P. Dogga, K. Narasimhan, A. Sivaraman, S.K. Saini, G. Varghese, and R. Netravali, "Revelio: Ml-generated debugging queries for distributed systems," arXiv preprint arXiv:2106.14347, 2021.
[5] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, D. Liu, Q. Xiang, and C. He, "Latent error prediction and fault localization for microservice applications by learning from system trace logs," Proc. 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp.683–694, 2019.
[6] 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.
[7] D. Liu, C. He, X. Peng, F. Lin, C. Zhang, S. Gong, Z. Li, J. Ou, and Z. Wu, "MicroHECL: High-efficient root cause localization in large-scale microservice systems," 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), pp.338–347, IEEE, 2021.
[8] L. Mariani, C. Monni, M. Pezzé, O. Riganelli, and R. Xin, "Localizing faults in cloud systems," 2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST), pp.262–273, IEEE, 2018.
[9] M. Li, D. Tang, Z. Wen, and Y. Cheng, "Microservice anomaly detection based on tracing data using semi-supervised learning," 2021 4th International Conference on Artificial Intelligence and Big Data (ICAIBD), pp.38–44, IEEE, 2021.
[10] L. Meng, F. Ji, Y. Sun, and T. Wang, "Detecting anomalies in microservices with execution trace comparison," Future Generation Computer Systems, vol.116, pp.291–301, 2021.
[11] H. Mi, H. Wang, Y. Zhou, M.R.-T. Lyu, and H. Cai, "Toward fine-grained, unsupervised, scalable performance diagnosis for production cloud computing systems," IEEE Trans. Parallel Distrib. Syst., vol.24, no.6, pp.1245–1255, 2013.
[12] Y. Gan, Y. Zhang, K. Hu, D. Cheng, Y. He, M. Pancholi, and C. Delimitrou, "Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices," Proc. Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, pp.19–33, 2019.
[13] Y. Gan, S. Dev, D. Lo, and C. Delimitrou, "Sage: Leveraging ML to diagnose unpredictable performance in cloud microservices," ML for Computer Architecture and Systems, 2020.
[14] P. Liu, H. Xu, Q. Ouyang, R. Jiao, Z. Chen, S. Zhang, J. Yang, L. Mo, J. Zeng, W. Xue, and D. Pei, "Unsupervised detection of microservice trace anomalies through service-level deep Bayesian networks," 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), pp.48–58, IEEE, 2020.
[15] J. Thalheim, A. Rodrigues, I.E. Akkus, P. Bhatotia, R. Chen, B. Viswanath, L. Jiao, and C. Fetzer, "Sieve: Actionable insights from monitored metrics in distributed systems," Proc. 18th ACM/IFIP/USENIX Middleware Conference, pp.14–27, 2017.
[16] P. Chen, Y. Qi, P. Zheng, and D. Hou, "CauseInfer: Automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems," IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, pp.1887–1895, IEEE, 2014.
[17] P. Spirtes, C.N. Glymour, R. Scheines, and D. Heckerman, Causation, Prediction, and Search, MIT Press, 2000.
[18] 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," 2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS), pp.1–10, IEEE, 2020.
[19] L. Wu, J. Tordsson, J. Bogatinovski, E. Elmroth, and O. Kao, "MicroDiag: Fine-grained performance diagnosis for microservice systems," 2021 IEEE/ACM International Workshop on Cloud Intelligence (CloudIntelligence), pp.31–36, IEEE, 2021.
[20] M. Müller, "Dynamic time warping," Information Retrieval for Music and Motion, pp.69–84, Springer, 2007.
[21] 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," 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE), pp.91–102, 2021.
[22] "International aiops challenge," http://iops.ai
[23] https://mp.weixin.qq.com/s/dJqOcSwkvyCHITCUjiwzAQ
[24] L. Wu, J. Tordsson, E. Elmroth, and O. Kao, "MicroRCA: Root cause localization of performance issues in microservices," Proc. IEEE/IFIP Network Operations and Management Symposium, pp.1–9, 2020.
[25] W. Cao, Y. Gao, B. Lin, X. Feng, Y. Xie, X. Lou, and P. Wang, "TcpRT: Instrument and diagnostic analysis system for service quality of cloud databases at massive scale in real-time," Proc. 2018 International Conference on Management of Data, pp.615–627, 2018.
[26] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang, "Time-series anomaly detection service at Microsoft," Proc. 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp.3009–3017, 2019.

Jingjing Yang received her B.S. degree from Beijing Jiaotong University in 2013. She is currently pursuing a Ph.D. degree in communication and information systems at Beijing Jiaotong University. Her main research interests are resource management of edge computing and performance diagnosis of microservice systems.

Yuchun Guo is a professor at Beijing Jiaotong University. Her research interests lie in information networks, QoE, privacy, and network intelligence. She received her M.S. degree from Beijing Normal University in 1993, and her Ph.D. degree from Beijing Jiaotong University in 2008. She spent an academic year at the Delft University of Technology, the Netherlands, from 2001 to 2002, as a visiting scientist.



Yishuai Chen received his B.S., M.S., and Ph.D. degrees from the School of Electronics and Information Engineering at Beijing Jiaotong University, in 1998, 2001, and 2010, respectively. He is currently an Associate Professor at the School of Electrical and Information Engineering of Beijing Jiaotong University. He has served as a TPC member for IEEE GLOBECOM.