Root-Cause Metric Location for Microservice Systems via Log Anomaly Detection

Lingzhi Wang[‡][||][†], Nengwen Zhao[||][†], Junjie Chen[‡][*], Pinnong Li[||], Wenchi Zhang[¶], Kaixin Sui[¶]
[‡]College of Intelligence and Computing, Tianjin University [||]Tsinghua University [¶]BizSeer

***Abstract*—Microservice systems are typically fragile and failures are inevitable in them due to their complexity and large scale. However, it is challenging to localize the root-cause metric due to its complicated dependencies and the huge number of various metrics. Existing methods are based on either correlation between metrics or correlation between metrics and failures. All of them ignore the key data source in microservice, i.e., logs. In this paper, we propose a novel root-cause metric localization approach by incorporating log anomaly detection. Our approach is based on a key observation, the value of root-cause metric should be changed along with the change of the log anomaly score of the system caused by the failure. Specifically, our approach includes two components, collecting anomaly scores by log anomaly detection algorithm and identifying root-cause metric by robust correlation analysis with data augmentation. Experiments on an open-source benchmark microservice system have demonstrated our approach can identify root-cause metrics more accurately than existing methods and only require a short localization time. Therefore, our approach can assist engineers to save much effort in diagnosing and mitigating failures as soon as possible.**

I. Introduction

With the soaring increase of utilization of cloud infrastructure and large-scale systems, microservice architecture has been widely used by more and more industrial companies because of its independence to deploy and update [1]. Although tremendous efforts have been devoted to guaranteeing the service quality, microservice systems are typically fragile and failures are inevitable in them due to their complexity and large scale, which would cause huge economic loss or damage user experience. For example, the loss of one-hour downtime for `Amazon.com` on Prime Day in 2018 (its biggest sale event of the year) is up to \$100 million[1]. As a result, once a failure happens, it is in an urgent need to locate the root cause and mitigate the failure as soon as possible.

In general, in order to grasp the running status of the system in real time, monitoring system will collect numerous metrics (e.g., CPU utilization and network delay) and logs from each microservice component. When the system fails, we need to identify the root-cause metric, which is able to reflect the root cause of the failure to a large degree. For example, when a failure occurs due to limited resources, it would be helpful to diagnose this failure by identifying the CPU utilization of the microservice as the root-cause metric. However, numerous metrics will behave abnormally due to complicated dependencies and failure propagation among components. For example, given a system consisting of 2 microservices shown in Fig. 1, they interact with each other by sending requests and messages. As each microservice has complex invocation chains, a delay of network transmission in microservice A will cause an irregular decrease of CPU utilization and response rate of another service B, since B needs to get the requests from A to conduct its own work. For A itself, some metrics such as network velocity and time-delay may be influenced by this failure. It is difficult to locate which microservice is the troublemaker considering this propagation, much less to find out the type of the failure. Besides, the failure propagation is so quick that we cannot identify the root-cause metric by time lag [2]. Therefore, it is challenging to locate the root-cause metric from a mass of metrics.

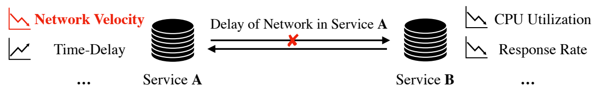


Fig. 1: An intuitive example to illustrate failure propagation between two microservices.

In the literature, a great number of efforts have been devoted to diagnose the failure and locate the root cause [3], [4], [5], [6], [7]. All of these related works focus on either correlation between metrics [3], [4] or correlation between metrics and failures (events) [5], [6], [7]. However, all of them ignore to make full use of other data sources in the monitoring system, for example, system logs which record detailed running information of each microservice. More specifically, over the years, a large body of research efforts have been dedicated to log anomaly detection and current log anomaly detection has achieved great effectiveness [8], [9]. Therefore, the technique of log anomaly detection can be leveraged to obtain the abnormal degree (also called anomaly score) of each microservice. If we can find a metric whose values have a large correlation with the anomaly scores acquired from log anomaly detection, this metric has the large possibility of being the root-cause one. As far as we know, most system exceptions in reality can be observed in both system logs and monitoring indicators, including three types we introduced in IV. That is, our key insight is to borrow the accumulated power of log anomaly detection to facilitate the identification of the root-

∗ Corresponding author.
† Equal contribution.
[1]https://www.businessinsider.com/amazon-prime-day-website-issues-cost-it-millions-in-lost-sales-2018-7

cause metric.

With this insight, we propose our approach, mainly consisting of two components *Collecting Anomaly Scores* and *Identifying Root-Cause Metric*. In detail, we first adopt the state-of-the-art log anomaly detection algorithm, DeepLog [8], to obtain the anomaly scores of the system. Afterwards, due to the data imbalance between normal time and abnormal time, we conduct data augmentation (oversampling and adding noise) for robust correlation analysis based on Mutual Information. Finally, our approach provides a root-cause metric ranking list for engineers based on the correlation results.

To evaluate the effectiveness of our proposed approach, we conduct experiments based on a widely-used microservice benchmark system named TrainTicket, which contains more than 30 microservices [1]. We inject three types of failures on the benchmark system, i.e, computing resources exhaustion, network transmission delay, and network transmission abortion. Our approach can identify the root-cause metric on top-15 on average among hundreds of metrics, which outperforms existing methods in terms of both effectiveness and efficiency.

To sum up, our work has the following major contributions:

- We propose a novel root-cause metric localization approach for failure diagnosis by incorporating log anomaly detection.
- We creatively leverage the technique of data augmentation for robust correlation analysis, which has been demonstrated to be effective.
- We conduct experiments based on a widely-used benchmark system, demonstrating that our proposed approach can identify the root-cause metric accurately and efficiently compared with other baseline methods.

II. Background

In this section, we first introduce some background about microservice briefly. Then two kinds of key data sources, i.e., metrics and logs are introduced.

A. Microservice Architecture

Microservice architecture is a variant of the service-oriented architecture (SOA) structural style that organizes a system as many light-weighted, loosely-coupled, independently deployed services and each is called a microservice [10]. Each microservice is flexible, being able to be implemented using different programming languages, database structures and environments. Therefore, rather than being a layer within a monolithic application like traditional architecture, microservice is suitable for a continuous processing of software development. People could adjust the number of microservices to meet the requirement of business [11]. It has become prevalent in the industrial and business area, used by many Internet companies such as Amazon, Google and Netflix.

B. Data Description

1) Metrics: Metric is a kind of time series data, having a format of (timestamp, value) [12], collected from many data sources, including but are not limited to network traces,

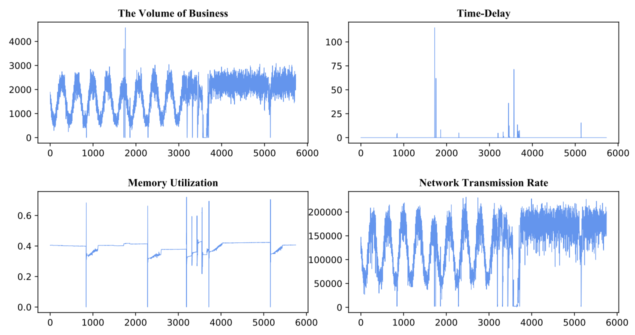
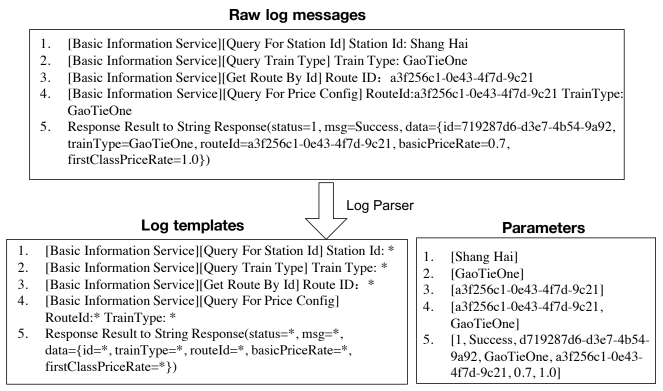


Fig. 2: Some typical metrics collected from TrainTicket.



```
1.  [Basic Information Service][Query For Station Id] Station Id: Shang Hai
2.  [Basic Information Service][Query Train Type] Train Type: GaoTieOne
3.  [Basic Information Service][Get Route By Id] Route ID： a3f256c1-0e43-4f7d-9c21
4.  [Basic Information Service][Query For Price Config] RouteId:a3f256c1-0e43-4f7d-9c21 TrainType:
    GaoTieOne
5.  Response Result to String Response(status=1, msg=Success, data={id=719287d6-d3e7-4b54-9a92,
    trainType=GaoTieOne, routeId=a3f256c1-0e43-4f7d-9c21, basicPriceRate=0.7,
    firstClassPriceRate=1.0})
```

```
1.  [Basic Information Service][Query For Station Id] Station Id: *
2.  [Basic Information Service][Query Train Type] Train Type: *
3.  [Basic Information Service][Get Route By Id] Route ID： *
4.  [Basic Information Service][Query For Price Config]
    RouteId:* TrainType: *
5.  Response Result to String Response(status=*, msg=*,
    data={id=*, trainType=*, routeId=*, basicPriceRate=*,
    firstClassPriceRate=*})
```

```
1.  [Shang Hai]
2.  [GaoTieOne]
3.  [a3f256c1-0e43-4f7d-9c21]
4.  [a3f256c1-0e43-4f7d-9c21,
    GaoTieOne]
5.  [1, Success, d719287d6-d3e7-4b54-
    9a92, GaoTieOne, a3f256c1-0e43-
    4f7d-9c21, 0.7, 1.0]
```

Fig. 3: An intuitive example to illustrate log parsing.

system logs and monitoring systems such as Prometheus[2]. Metrics used to characterize the status of systems can be usually divided into two categories, business-level metrics (e.g., transaction volume) and machine-level metrics (e.g., CPU utilization). Fig. 2 presents some representative metrics collected from the benchmark system used in our experiments.

2) Logs: Log is one of the most valuable data sources in microservice. Compared with metrics, logs record more detailed running information about the system as well as important activities of users and events of interaction with hardware. Usually, logs are generated using the "print" function with a string template and detailed information as parameters. Thus logs are unstructured or semi-unstructured texts, which brings challenges to log analysis. Typically, these logs should be properly parsed to extract the template and parameters for further analysis. Fig. 3 presents an intuitive example to illustrate log parsing. Many data-driven methods have been proposed for log parsing [13], [14], such as SLCT[15], LogSig[16] and IPLoM[17]. In our approach, we use Drain proposed in [18] to extract log templates, which has shown better performance both on accuracy and efficiency [18].

III. Approach

A. Overview

In this paper, we propose a novel approach to identifying the root-cause metric by incorporating log anomaly detection. The core idea is that a metric whose values have a large correlation

[2]https://prometheus.io/

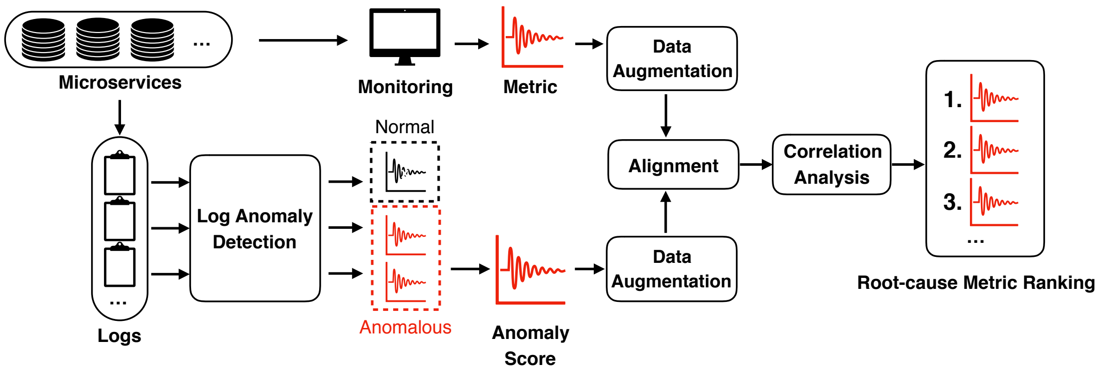


Fig. 4: The overall framework of our approach.

with the anomaly scores acquired from log anomaly detection, has the large possibility of being the root-cause one.

Specifically, our approach shown in Fig. 4 has two key steps, i.e., *Collecting Anomaly Scores* and *Identifying Root-Cause Metric*. First, we make use of the state-of-the-art log anomaly detection algorithm, DeepLog [8], to obtain the anomaly score of the microservice system, which can be used to characterize the abnormal degree of the system. Then we conduct the correlation analysis based on Mutual Information [19] to identify the root-cause metric. Before doing that, we first need to align anomaly score and monitoring metrics. However, considering the data imbalance existing in practice (normal time is much longer than abnormal time), we adopt two data augmentation methods to enhance the correlation analysis, so as to obtain better results. Finally, our approach can provide a ranking list based on the correlation score, which means the probability ranking of each metric to be the root cause.

B. *Collecting Anomaly Scores*

As mentioned earlier, the existing root-cause metric localization approaches are based on either correlation between metrics or correlation between metrics and failures. However, we novelly propose to make use of system logs and get system anomaly scores via log anomaly detection, which requires a chronologically ordered log sequence for every microservice. In this way, we can localize the root-cause metric based on the correlation between anomaly scores and metrics, which is more comprehensive than existing methods.

In the literature, significant research efforts have been devoted to log anomaly detection in the field of security and system reliability, for example, clustering-based method [9], Principle Component Analysis (PCA) [20], and invariant mining [21]. In our approach, we adopt DeepLog [8], the state-of-the-art log anomaly detection algorithm, which has shown better performance compared with other methods. In detail, DeepLog detects log anomalies from two perspectives. One is the anomalous log template sequences and the other is anomalous log parameters (log templates and log parameters have been introduced in Section II-B2). Due to the limit of space, more details about DeepLog can be found in [8].

In our approach, we directly adopt the log templates anomaly detection in DeepLog. About the log parameter anomaly detection, considering the large number of and the large variety of log parameters, it is meaningless and time-consuming to detect parameter anomalies for each kind of log template, since the majority of parameters are strings instead of numbers, which cannot be represented as time series. Thus we only record the time interval between two logs and regard the time interval as a kind of parameter, which also mentioned in DeepLog [8]. Time interval anomaly detection is intuitive based on one key observation: if a microservice does not receive logs for a long time, it is likely to be abnormal. Therefore, for each log l, DeepLog can output an anomaly score $AS(l)$ based on Eq.1, where $AS_t(l)$ and $AS_p(l)$ are anomaly score of template sequences and time interval, respectively. The value of weight (w) is set to 0.5 in our experiments.

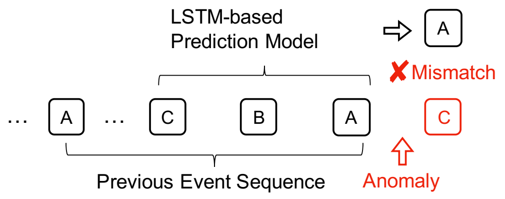
$$AS(l) = w \times AS_t(l) + (1 - w) \times AS_p(l) \qquad (1)$$

Finally, for each microservice, we can obtain a series of anomaly scores. Based on the threshold given by engineers, we can find the abnormal microservices and filter the root-cause metric generated from the normal microservice. Finally, we get a series of anomaly scores representing the abnormal degree of the whole microservice system by computing the mean value of all abnormal anomaly scores.

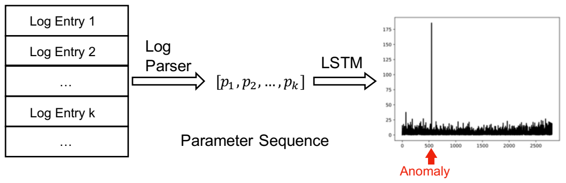
C. *Identifying Root-Cause Metric*

Intuitively, if we can find a metric whose values have a higher correlation with the anomaly scores acquired from log anomaly detection, this metric has a larger possibility of being the root-cause one. Motivated by this observation, we identify the root-cause metric based on the correlation analysis between log anomaly scores and metrics.

1) Data Augmentation: Due to the imbalance between normal data and abnormal data, directly applying correlation analysis performs not well in our experiments. Therefore, we leverage the idea of data augmentation, which has been widely used in deep learning and computer vision [22], to overcome the data imbalance problem. In detail, we adopt



(a) The process of detecting anomalous log template sequences



(b) The process of detecting anomalous log parameters sequences

Fig. 5: Two types of anomalies detected by DeepLog

the following tricks, i.e., oversampling anomalies and adding Gaussian noise.

Oversampling anomalies. Intuitively, because of data imbalance (the normal time is much longer than abnormal time), the correlation during normal time will account for a large part, which will weaken the key correlation during the abnormal time. However, what we really care about is the behavior of this metric when the system failed. The most direct way of reducing the negative influence caused by normal points is simply increasing the number of anomalous points in the time series. In our approach, we oversample the anomalous points by expanding the length of abnormal time. Specifically, if the abnormal length is L, we expand it into $(1+\alpha)L$ and $\alpha = 0.3$ in our experiments. In Section IV-D1, we will further discuss the influence of different expanded lengths (α) on the results.

Adding Gaussian noise. Adding noises is a popular method for data augmentation, which can enhance the robustness of the model and avoid overfitting effectively. Therefore, in our approach, we add Gaussian noises in the log anomaly score series and metrics. Besides, the values of noise intensity (the variance of the Gaussian distribution) have an influence on results, and we will discuss it in detail in Section IV-D1.

The above two tricks on data augmentation can significantly improve the performance of correlation analysis, which will also be demonstrated in Section IV-D1.

2) Correlation Analysis: After data augmentation, we conduct the correlation analysis between log anomaly score series and each metric, so as to locate the root cause. In the field of machine learning and statistical analysis, a large number of correlation analysis methods have been proposed [23], [24], for example, Pearson Coefficient, Kendall Coefficient, Spearman Coefficient, Maximal Information Coefficient and Mutual Information. Based on our observations and experiments in Section IV-D2, we find that Mutual Information (MI) achieves the best performance among these alternatives because of its capability of catching non-linear dependency and adaptation to various data. Besides, MI does not make any assumptions about some attributes of the variables, such as normal distribution. Therefore, we adopt Mutual Information in our approach.

Mutual Information is commonly used to evaluate the mutual dependence between two variables, which is initially applied in the area of communication technology. More specifically, let (X, Y) be a pair of random variables, the Mutual Information between them is defined as:

$$I(X, Y) = H(Y) - H(Y|X) \tag{2}$$

where $H(Y)$ represents the uncertainty of variable Y, which was defined by Shannon as entropy with the following mathematical form:

$$H(Y) = \int_y (p(y)) \log (p(y)) \tag{3}$$

Similarly, the conditional entropy $H(Y|X)$ is defined as:

$$H(Y|X) = \iint_{x,y} (p(x, y)) \log (p(y|x)) \tag{4}$$

Compared to Pearson Coefficient, MI can evaluate both linear dependence and non-linear dependence, which makes it more suitable for our problem. Calculating MI directly according to its definition requires us to know the Probability Density Function (PDF) of the variables we are interested in. It is difficult because we only have finite samples in most cases. [25] and [26] provide some novel methods to estimate the probability density based on entropy estimation from k-nearest neighbor distances. In this way, we can calculate MI between the log anomaly score and each metric. Finally, our approach can provide a metric ranking list based on the correlation scores. According to our insight, a metric whose values have a larger correlation with the anomaly scores acquired from log anomaly detection, has a larger possibility of being the root cause. Therefore, the metrics located on the top result are more likely to be the root cause, so that engineers can examine each metric one by one based on the result.

IV. Evaluation

In this section, we aim to evaluate the performance of our proposed approach and answer the following research questions (RQs):

- RQ1: Is our approach effective in root-cause metric localization compared with other baseline methods?
- RQ2: Does each component contribute (data augmentation and correlation analysis) to our approach?
- RQ3: What is the time efficiency of our approach?

A. Experiment Setup

We evaluate the performance of our approach based on a medium-size open-source benchmark microservice system named TrainTicket [27]. This platform serves as a system of selling train tickets, which is based on microservice architecture, containing 41 microservices including pay, price, order, food, etc. Besides, as guided in [1], we can inject different kinds of failures manually to see if we can locate the root-cause metric accurately based on the logs and metrics collected from the benchmark system. In our evaluation, we implement the injection of three types of failures.

Computing Resources Exhaustion. Stress-ng[3] is a widely-used tool to test the capability of a computer system to deal with diverse types of stress. By running stress-ng in a designative docker, the resource of computing may be extremely scarce because of the existence of an extra process which requires a large amount of computing resource. Therefore, system running must be influenced. Fig. 6 presents some abnormal metrics caused by computing resources exhaustion injected in the food microservice, including CPU utilization, memory utilization, time-delay and successful transaction rate.

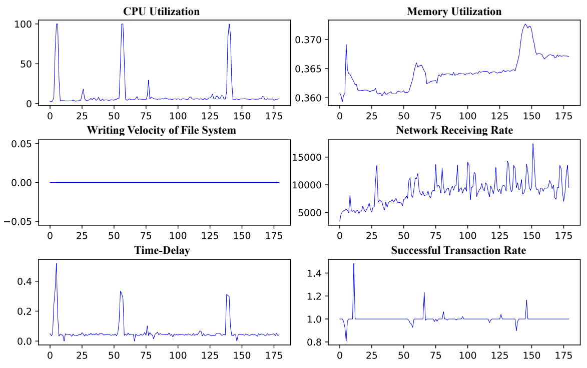


Fig. 6: The behavior of metrics influenced by CPU exhaustion.

Network Transmission Delay. We inject HTTP delay failure by istio[4], which is designed to connect all microservices in a more reliable and secure way. Intuitively, the network delay will bring troubles to TrainTicket and some metrics, like network transmit/receive rate.

Network Transmission Abortion. We also inject HTTP abort failure by istio. In this case, the faulted microservice cannot communicate with other components and network-related metrics will be affected.

B. Evaluation Metrics

Intuitively, the output of our approach is the ranking list of root-cause metrics, thus we adopt some popular metrics used in ranking problem to evaluate the performance of our approach. Specifically, we choose to use the top-k precision and average ranking as evaluation metrics in our experiments. Besides, we also take the running time of our approach into consideration.

C. Performance of Root-cause Metric Localization

1) Baseline: We compare the effectiveness of our approach on root-cause metric localization with three related baselines.

[3]https://kernel.ubuntu.com/ cking/stress-ng/
[4]https://istio.io/

ϵ-Diagnosis. The core idea of ϵ-Diagnosis [5] is to take the metric during the adjacent period of the failure, and conduct ϵ-statistics test based on energy distance with the normal metric. The larger the statistical gap is, the more critical the metric is to explain the failure. The approach is based on a premise that the true root-cause metric must change drastically when a failure occurs. In our problem, we use log anomaly score to detect anomalies for ϵ-Diagnosis. We use sliding window to get subsequences from a metric and maintain two sets, normal subsequences set S_N and abnormal subsequences set S_A for each metric. Two-sample test is adopted to help us determine whether the two sets are different statistically. ϵ-Diagnosis uses energy distance to calculate the distance between S_A and S_N:

$$\rho^2(S_A, S_N) = \begin{cases} \frac{cov^2(S_A,S_N)}{\sqrt{\sigma^2(S_A)\sigma^2(S_N)}}, & \sigma^2(S_A)\sigma^2(S_N) > 0 \\ 0, & \sigma^2(S_A)\sigma^2(S_N) = 0 \end{cases} \tag{5}$$

Regression-based Analysis: Regression-based localization method [6] assumes a latent linear relationship lies in anomaly scores and metrics. That is to say, in our problem, given the log anomaly score (y) as the dependent variable and all metrics $x_1, \cdots, x_n$, we assume that they satisfy the following equation:

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon \tag{6}$$

where ϵ is the residual error of this model. The coefficients $(\beta_1, \cdots, \beta_n)$ quantify the contribution of a metric to anomaly score. The larger β_k is in absolute terms, the more significant the k-th metric is, and we have more confidence to assert that the k-th metric is root cause. As introduced in [6], lots of regression models can be used, such as Possion regression and ridge regression, etc., depending on the data distribution. We choose ordinary least squares (OLS) used in [6], to determine the importance of metrics by comparing the absolute values of coefficients.

Correlating Events with Time Series: Similar to ϵ-Diagnosis, both of them assume that if an event has a relationship with a time series, then this time series must change when the event occurs [7]. Different from the ϵ-statistic, [7] utilizes the nearest neighbor statistic to evaluate the difference between two sets. Given the normal subsequence set $\mathbf{S}_N$ and the abnormal subsequence set $\mathbf{S}_A$, the first thing we need to do is to construct $\mathbf{S}_T = \mathbf{S}_A \cup \mathbf{S}_N$. For S_t in set $\mathbf{S}_T$, We use $NN_r(St, \mathbf{S}_T)$ to represent the rth nearest neighbor of S_t in the set $\{\mathbf{S}_T \backslash S_t\}$. Then we calculate

$$I_r(S_t, \mathbf{S}_A, \mathbf{S}_N) = \begin{cases} 1, & if S_t \in \mathbf{S}_i and NN_r(St, \mathbf{S}_T) \in \mathbf{S}_i \\ 0, & otherwise \end{cases} \tag{7}$$

The indicative function just shows if the nearest neighbor of S_t and S_t itself are in the same subset. Then we can construct the following statistic

$$T_{r,p} = \frac{1}{pr} \sum_{i=1}^{p} \sum_{j=1}^{r} I_j(S_t, \mathbf{S}_A, \mathbf{S}_N) \tag{8}$$

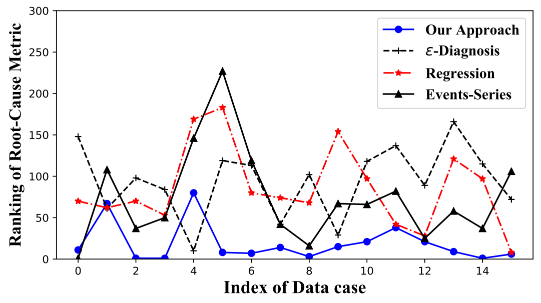


Fig. 7: Ranking of root-cause metric in different failure cases.

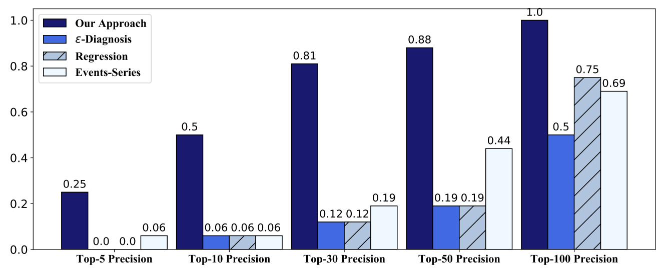


Fig. 8: Top-k precision comparison between our approach and baseline methods.

TABLE I: Comparison of average ranking of root-cause metric between different methods.

Methods	Average Ranking of Root-Cause Metric
Our Approach	14.375
ϵ-Diagnosis	93.9375
Regression	86
Event-series	74.1875

TABLE II: Effectiveness of data augmentation.

Metric	Data Augmentation	Without Augmentation
Average Ranking	14.375	19.875
Top-5	0.25	0.13
Top-10	0.56	0.38
Top-30	0.94	0.75
Top-50	0.94	0.94
Top-100	1.00	1.00

where $p = |\mathbf{S}_A| + |\mathbf{S}_N|$. From the formula of $T_{r,p}$ we could know that the larger the statistic $T_{r,p}$ is, the more $I_r(St, \mathbf{S}_A, \mathbf{S}_N)$ are equal to 1, which means the two sets are more heterogeneous based on the distance measurement we choose. [28] shows more details from a mathematical point of view. Many distance measurements could be utilized to calculate $NN_r(St, \mathbf{S}_T)$. In our experiments, we choose the Euclidean distance because of its lower computing cost.

2) Results: We evaluate the effectiveness of root-cause metric localization of our approach and three baseline methods on 16 failure cases. Fig. 7 presents the ranking of the root-cause metric in these 16 failure cases. Intuitively, we can observe that the root-cause metric ranks at or near the top using our approach in most cases. However, the three baseline methods fail to locate the root-cause metric in the top ranking list, which means that engineers need to spend much time in finding the true root cause. In order to present the effectiveness of our approach from a global view, we calculate the average ranking of the root-cause metric on different cases and the results are shown in Table I. From this table, we can observe that our approach can locate the root-cause metric on top-15 on average from hundreds of metrics, which performs much better than baseline methods.

Furthermore, we also investigate the top-k precision in our evaluation. If the top-k precision is high, it means that engineers can find the root-cause metric in a short time and do not waste much time on these false negatives. Fig. 8 presents the top-k precision ($k = 5, 10, 30, 50, 100$) comparison between our approach and three baseline methods. We can clearly observe our approach can achieve the best top-k precision continuously, which can save much effort for engineers.

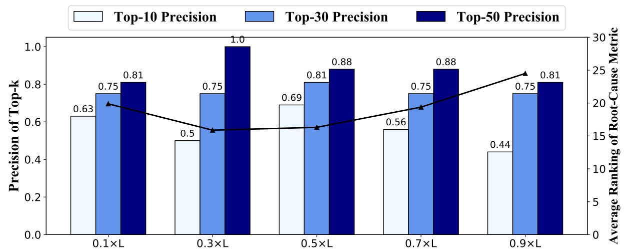
In conclusion, by incorporating log anomaly detection to obtain the abnormal degree of the system, our approach outperforms three baseline methods and can locate the root-cause metric at top-15 on average from hundreds of metrics.

D. Contribution of Each Component

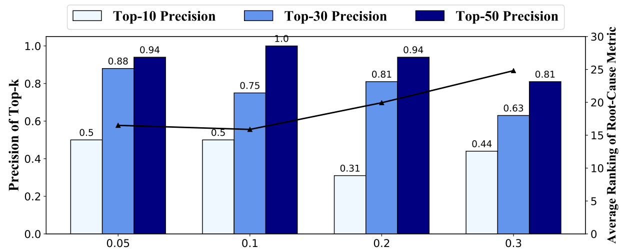
In this RQ, we aim to evaluate the effectiveness of two key techniques, i.e., data augmentation and correlation analysis in our approach.

1) Data Augmentation: As introduced in Section III-C1, to overcome the challenge of data imbalance between normal time and abnormal time, we conduct data augmentation to ensure robust correlation analysis. In order to demonstrate the necessity of adopting data augmentation, Table II presents the comparison results (average ranking and top-k precision) between with and without data augmentation. From this table, we can observe data augmentation is indeed effective and can significantly improve the performance. Specifically, data augmentation can improve the average ranking from 19.875 to 14.375 and increase the top-k ($k = 5, 10, 30$) precision by about 0.2.

Furthermore, we also investigate the influence of different parameters in data augmentation on localization performance. There are two parameters which are crucial to data augmentation, oversampling ratio and noise intensity. First, about oversampling ration, as introduced in Section III-C1, we expand the length of abnormal time to oversample abnormal data and the value of α denotes the oversampling ration. Fig. 9(a) shows the average ranking and top-k precision under different oversampling ratios ($\alpha = 0.1, 0.3, 0.5, 0.7, 0.9$). We can observe that our approach achieves the best performance when oversampling ration is equal to 0.3. Then about the noise intensity, Fig. 9(b) shows the average ranking and top-k precision under different noise intensities (i.e., the variance of Gaussian distribution). It is obvious that our approach performs best when the variance is equal to 0.1. Intuitively, if the noise intensity is too small, it has no obvious effect.



(a) Comparison between different oversampling ratios



(b) Comparison between different noise intensities

Fig. 9: Influences of different data augmentation parameters on performance.

TABLE III: Effectiveness of correlation analysis.

Metric	MI	MIC	Pearson
Average Ranking	14.375	29.813	20.563
Top-5	0.38	0.19	0.19
Top-10	0.50	0.44	0.44
Top-30	0.75	0.69	0.88
Top-50	1.00	0.69	0.88
Top-100	1.00	0.94	1.00

However, with too large intensity, the raw information of the data will be destroyed, which also leads to unsatisfactory results.

In conclusion, data augmentation adopted in our approach is simple but indeed effective.

2) Correlation Analysis: Correlation analysis between log anomaly score and each metric is also a key component in our approach. In Section III-C2, we have mentioned there are many correlation analysis methods in the literature and we choose to adopt Mutual Information (MI) in our approach. In order to demonstrate the effectiveness of MI, we compare MI with another two popular correlation analysis methods, i.e., Maximal Information Coefficient (MIC) and Pearson Coefficient. The compared average ranking and top-k precision are shown in Table III. Obviously, Mutual Information indeed can achieve the best performance compared with the other two correlation analysis alternatives. In conclusion, the correlation analysis method adopted in our approach is indeed effective.

E. Time Efficiency

In addition to the ranking precision, efficiency is another important evaluation metric we need to consider. It is because when a failure happens in practice, it is vital to locate the root cause as soon as possible, so that engineers can take immediate actions to mitigate the failure and reduce the MTTR (mean time to repair). The running time comparison between our approach and three baselines is demonstrated in Fig. 10.

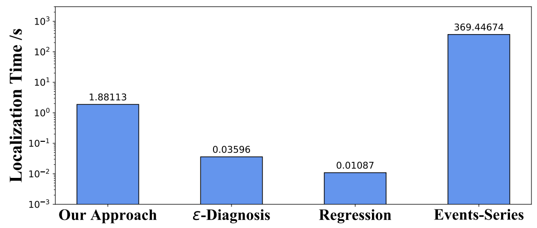


Fig. 10: Time efficiency comparison between our approach and baseline methods.

Obviously, our approach can achieve a short running time to localize the root-cause metric (about 1.8 seconds). Actually, log anomaly detection will cost much time. However, the DeepLog model can be trained offline with historical data and anomaly detection can continuously run with online incoming log data, so that we can obtain the anomaly score directly, which can save much time for root-cause metric localization. In terms of the baselines, both $\epsilon-$Diagnosis and regression-based analysis require less than 0.05s on average, but their localization accuracy are far from satisfying. In terms of correlating events with time series, it requires hundreds of seconds to localize the root-cause metric. It means that engineers need to wait a few minutes to get a not very accurate result, which will result in unavailability in real world. As a result, take both accuracy and efficiency into consideration, our approach is the best choice for root-cause metric localization.

V. Related Work

A. Root-cause Metric Localization

A great deal of efforts have been devoted into root-cause metric localization. These methods can be divided into two categories, metric-event correlation and metric-metric correlation. About metric-event correlation, the majority of methods utilize hypothesis test. [7] adopts nearest neighbor statistic to test if there was any change in metrics after an event had occurred. [5] makes use of ϵ-statistics to measure the difference. [6] transfers the event sequence into another time sequence, and uses the regression model to evaluate causalities between them. These approaches just regard the failure as a single event, instead of mining the detailed information about the abnormal degrees from system logs in depth, which restricts their performance as we discussed before.

In terms of metric-metric correlation based methods, time lag between two metrics has been widely used for root-cause metric localization [2]. It is based on a key observation that root-cause metric is always behave abnormally earlier than other influenced metrics. However, in practice, the metrics are usually collected at every minute or even longer (e.g., five minutes), but failure propagates very quickly. Therefore, it is unrealistic to identify root-cause metric based on time lag between two metrics.

B. Log Anomaly Detection

In our approach, we leverage the technique of log anomaly detection to obtain the abnormal degree of the microservice system, so as to achieve a better performance of root-cause metric localization. In the literature, much efforts have been dedicated in log anomaly detection, for example, PCA-based method [20], clustering-based method [29], Invariant Mining [21]. However, DeepLog based on deep learning has shown better performance compared with other traditional methods. Therefore, considering the trade-off between the detection accuracy and the computing cost, we adopt DeepLog to obtain the anomaly score of the system.

VI. Conclusion

Root-cause metric localization is a challenging task due to the numerous metrics and complicated dependencies in practice. In this paper, we propose a novel and robust root-cause metric localization approach by incorporating log anomaly detection. Our approach consists of two parts, collecting anomaly scores by the state-of-the-art log anomaly detection algorithm and identifying root-cause metric by robust correlation analysis with data augmentation. Extensive experiments on a benchmark microservice system demonstrate our approach can localize the root-cause metric accurately compared with existing baseline methods and achieve a short response time. Therefore, our approach can assist engineers in saving much time and effort to diagnose and mitigate failures as soon as possible.

Acknowledgement

This work has been supported by the National Natural Science Foundation of China under Grant No. 61872263, U1836214, 61802275, 71502125, Intelligent Manufacturing Special Fund of Tianjin under Grant No. 20191012, Innovation Research Project of Tianjin University under Grant No. 2020XZC-0042.

References

[1] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, W. Li, and D. Ding, "Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study," *IEEE Transactions on Software Engineering*, 2018.

[2] C. Zeng, L. Tang, T. Li, L. Shwartz, and G. Y. Grabarnik, "Mining temporal lag from fluctuating events for correlation and root cause analysis," in *10th International Conference on Network and Service Management (CNSM) and Workshop*. IEEE, 2014, pp. 19–27.

[3] G. Li, S. J. Qin, and T. Yuan, "Data-driven root cause diagnosis of faults in process industries," *Chemometrics and Intelligent Laboratory Systems*, vol. 159, pp. 1–11, 2016.

[4] B. Rashidi, D. S. Singh, and Q. Zhao, "Data-driven root-cause fault diagnosis for multivariate non-linear processes," *Control Engineering Practice*, vol. 70, pp. 134–147, 2018.

[5] H. Shan, Y. Chen, H. Liu, Y. Zhang, X. Xiao, X. He, M. Li, and W. Ding, "?-diagnosis: Unsupervised and real-time diagnosis of small-window long-tail latency in large-scale microservice platforms," in *The World Wide Web Conference*. ACM, 2019, pp. 3215–3222.

[6] M. Farshchi, J.-G. Schneider, I. Weber, and J. Grundy, "Experience report: Anomaly detection of cloud application operations using log and cloud metric correlation analysis," in *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2015, pp. 24–34.

[7] C. Luo, J.-G. Lou, Q. Lin, Q. Fu, R. Ding, D. Zhang, and Z. Wang, "Correlating events with time series for incident diagnosis," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 1583–1592.

[8] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1285–1298.

[9] Q. Lin, H. Zhang, J.-G. Lou, and et al., "Log clustering based problem identification for online service systems," in *ICSE*. ACM, 2016, pp. 102–111.

[10] S. Newman, *Building microservices: designing fine-grained systems*. " O'Reilly Media, Inc.", 2015.

[11] C. Pautasso, O. Zimmermann, M. Amundsen, J. Lewis, and N. Josuttis, "Microservices in practice, part 1: Reality check and service design," *IEEE Software*, no. 1, pp. 91–98, 2017.

[12] D. Liu, Y. Zhao, H. Xu, Y. Sun, D. Pei, J. Luo, X. Jing, and M. Feng, "Opprentice: Towards practical and automatic anomaly detection through machine learning," in *Proceedings of the 2015 Internet Measurement Conference*. ACM, 2015, pp. 211–224.

[13] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "An evaluation study on log parsing and its use in log mining," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2016, pp. 654–661.

[14] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, "Tools and benchmarks for automated log parsing," in *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice*. IEEE Press, 2019, pp. 121–130.

[15] R. Vaarandi, "Mining event logs with slct and loghound," in *NOMS 2008-2008 IEEE Network Operations and Management Symposium*. IEEE, 2008, pp. 1071–1074.

[16] L. Tang, T. Li, and C.-S. Perng, "Logsig: Generating system events from raw textual logs," in *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 2011, pp. 785–794.

[17] A. Makanju, N. Zincir-Heywood, and E. Milios, "Iplom: Iterative partitioning log mining," *Tech. Rep. CS-2009-07*, 2009.

[18] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, 2017, pp. 33–40.

[19] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 8, pp. 1226–1238, 2005.

[20] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 2009, pp. 117–132.

[21] J.-G. Lou, Q. Fu, S. Yang, J. Li, and B. Wu, "Mining program workflow from interleaved traces," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 613–622.

[22] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.

[23] S.-B. Cho and H.-H. Won, "Machine learning in dna microarray analysis for cancer classification," in *Proceedings of the First Asia-Pacific bioinformatics conference on Bioinformatics 2003-Volume 19*. Australian Computer Society, Inc., 2003, pp. 189–198.

[24] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[25] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical review E*, vol. 69, no. 6, p. 066138, 2004.

[26] B. C. Ross, "Mutual information between discrete and continuous data sets," *PloS one*, vol. 9, no. 2, p. e87357, 2014.

[27] X. Zhou, X. Peng, T. Xie, J. Sun, C. Xu, C. Ji, and W. Zhao, "Poster: Benchmarking microservice systems for software engineering research," in *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*. IEEE, 2018, pp. 323–324.

[28] M. F. Schilling, "Multivariate two-sample tests based on nearest neighbors," *Journal of the American Statistical Association*, vol. 81, no. 395, pp. 799–806, 1986.

[29] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *Proceedings*

of the 38th International Conference on Software Engineering Companion. ACM, 2016, pp. 102–111.