Research on Microservice Anomaly Detection Technology Based on Conditional Random Field

Wei Cao\textsuperscript{1,}\textsuperscript{a} Zhiying Cao\textsuperscript{2,}\textsuperscript{b} Xiuguo Zhang\textsuperscript{3,}\textsuperscript{c}

\textsuperscript{1}Dalian Maritime University
Dalian, China
\textsuperscript{2}Dalian Maritime University
Dalian, China
\textsuperscript{3}Dalian Maritime University
Dalian, China
\textsuperscript{a}jackwei@dlmu.edu.cn \textsuperscript{b}czysophy@dlmu.edu.cn \textsuperscript{c}zhangxg@dlmu.edu.cn

ABSTRACT: Recently, the existing microservice anomaly detection methods which mainly monitor the system metrics and analyze the system logs have low efficiency and consume much time. This paper proposes a microservice anomaly detection method based on Conditional Random Field (CRF). The method takes the collected system parameters such as CPU utilization, memory utilization, bandwidth occupancy as the characteristic values of the observation sequence, and labels the abnormal types of the sequences corresponding to the feature values, and then creates the microservice fault matrix. Operators can quickly retrieve the specific anomaly information of the microservice error according to the microservice fault matrix. Compared with the traditional Hidden Markov Model-based detection method, this method can find the abnormal sequence corresponding to the observation sequence in the global scope. The experimental results show that the method proposed by this paper can accurately find the faults in the microservice system, and the accuracy and recall rate are relatively high.

CCS Concepts
• Computing methodologies~Anomaly detection • Computing methodologies~Supervised learning by classification • Information systems ~ RESTful web services

1. INTRODUCTION
Microservice develops a single application as a set of small services, each running in its own process, communicating with each other through a lightweight mechanism. With container technology, the microservice architecture can be better implemented [1,2]. However, there are many intricate dependencies between services in the microservice architecture. In an actual production environment, service interruption or service delay may occur due to slow network connection, heavy resources, offline, etc., resulting in the failure of some crucial services, which eventually leads to the collapse of the entire microservice system.

2. RELATED WORK
In order to detect the microservice anomaly, many methods have been proposed by researchers. Ziyong Wang [3] et al. propose a model based on execution trajectory, which constructs the execution
trajectory or critical path of the service in the microservice system through the request and call relationship between the service components, and then use the appropriate technology to analyze and find out the cause of the failure. Although this method can detect system faults finely, it is intrusive to the code, and too many points of the insertion will affect the performance of the microservice system.

Jingmin Xu et al. [4] propose a log model through the lifecycle Kubernetes log problem diagnosis tool, which can easily find exception log entries and exception declaration items based on runtime logs and deployment statements. However, the model is limited to analyzing yaml files, without further consideration of factors such as code and network connectivity, and fails to reveal the relationship between failures.

Peifei Chen [5] et al. propose a black box reasoning system, which can automatically construct a two-layer causality diagram, and analyze the critical path in the causal map to find out the cause of system performance failure through a series of statistical methods. Although this model can reduce the cost of system anomaly detection, it can not adapt to the dynamic and real-time nature of microservice. Once the system is huge, the reasoning process of the model will be slow.

Zhichun Jia [6] et al. propose a probabilistic model, which based on the principle of Bayesian and multi-fault inference. The multi-fault logic inference technique is used to obtain candidate diagnostic solutions and Bayesian formula is used to calculate the posterior probability of candidate solutions. Finally, the model gets an optimal diagnosis by sorting these candidate solutions. The advantage of this model is that it can locate multiple faults at the same time, which better solves the uncertainty of state and behavior faults in the service application system. But the shortcoming of this model is also obvious, that is the heavy reliance on historical data makes the construction of the service execution matrix difficult.

In summary, there are many deficiencies in the current anomaly detection technology research. Therefore, this paper proposes a microservice anomaly detection model based on conditional random field, which will start from the following aspects:

1. The granularity of anomaly detection is improved through fine microservice abnormality classification and anomaly detection method;
2. Modeling the external observation state does not invade the code of the original system, ensuring the security of the system;
3. We use the characteristic parameters of the system to construct a sequence of features for the type of fault phenomenon directly. The difficulty of mining information is reduced compared with the log, and the diagnostic quality is also improved.

3. MICROSERVICE ANOMALY DETECTION MODEL

3.1 Microservice Anomaly Classification
Anomaly classification is the basic work for microservice anomaly detection. A meticulous and efficient classification enables the system to react quickly to specific anomaly, thereby improving system performance. Because the types of microservice anomaly are quite many and complex, it is not easy to find a standard classification to summarize these anomalies. Therefore, based on many papers [7-9], this paper presents a microservice anomaly classification tree, which is classified as shown in Figure 1.
Figure 1. Microservice anomaly classification tree.

3.2 Model Data Acquisition and Data Preprocessing

Before implementing microservice anomaly detection, relevant parameters that reflect the state of the system must be collected [10]. According to the characteristics of the microservice system, this paper runs the relevant microservice on the Kubernetes platform, and then uses Prometheus to monitor the instances of the microservice and collect relevant parameters. The relevant parameters are shown in Table 1:

| Acquisition area | Acquisition parameter                                      |
|------------------|------------------------------------------------------------|
| CPU              | CPU utilization                                            |
| Memory           | Memory utilization                                         |
| Service thread   | Total number of threads                                    |
|                  | Number of thread requests per unit time                    |
|                  | Total number of thread execution successes                 |
|                  | Thread execution timeout total                             |
|                  | Single thread execution time                               |
Since the number of feature parameters collected is large, the value of each feature parameter is also large, so the obtained data set is large. In this paper, the method which presented in paper[10] is used to reduce the dimension of the data, which effectively solves the problem of too large dimension of the data set and reduces a large amount of time overhead.

3.3 Conditional Random Field Anomaly Detection Model

3.3.1 Model Construction

Since the feature parameters can characterize the state of the system, the state of the system can be made up of a combination of a set of feature parameters. In this paper, a batch of data is collected in advance, and the collected feature parameters are regarded as observation sequences in time series. The anomaly species are marked for each feature, and recorded , and then the marker sequence is obtained. Because the above mentioned is a process of obtaining a labeled sequence from an observed sequence, this paper uses conditional random field (CRF) as a model for microservice anomaly detection [11].

As shown in Figure 2, this paper uses the linear chain CRF model for microservice anomaly detection. The mathematical formula for linear chain conditional random field is:

\[ P(y|x) = \frac{1}{Z(x)} \exp \sum_{i,k} \lambda_k f_k(y_{i+1}, y_i, x, i) + \sum_{i,j} \mu_j s_j(y_i, x, i) \]  \hspace{1cm} (1)

Among the formula (1),
\[ Z(x) = \sum_{y} \exp \sum_{i,k} \lambda_k f_k(y_{i+1}, y_i, x, i) + \sum_{i,j} \mu_j s_j(y_i, x, i) \]  \hspace{1cm} (2)

Among the formulas, \( f_k \) is a feature function defined on the edge, called a transition feature, depending on the current and previous positions. \( s_j \) is a feature function defined on a node, called a state feature, depending on the current position. \( \lambda_k \) and \( \mu_j \) are the weights corresponding to the feature function. \( Z(x) \) is a normalization factor, which is a necessary condition for converting the potential function product into a legal probability distribution. Both \( f_k \) and \( s_j \) depend on the location and belong to the local feature function. Since the same feature is defined at each position, this paper sums the same feature at each position and converts the local feature function into a global feature function [12].

![Figure 2. Linear chain CRF model.](image)

The transition feature, state feature and their weights are first represented by a uniform symbol. There are \( K_1 \) transfer features, \( K_2 \) state features. \( K = K_1 + K_2 \), and then

\[ f_k(y_{i+1}, y_i, x, i), k = 1, 2, \cdots, K_1 \]
\[ s_j(y_i, x, i), k = K_1 + l, l = 1, 2, \cdots, K_2 \]  \hspace{1cm} (3)

The transfer feature and the state feature are summed at each position \( i \), recorded as
\[ f_k(y, x) = \sum_{i=1}^{n} f_k(y_{i-1}, y_j, x_i), k = 1, 2, \ldots, K \]  

\( w_k \) represents the weight of the feature function \( f_k(y, x) \), ie

\[ w_k = \begin{cases} \lambda_k, & k = 1, 2, \ldots, K_1 \\ \mu_l, & k = K_1 + l, l = 1, 2, \ldots, K_2 \end{cases} \]  

Then the conditional random field can be expressed as

\[ P_y(y|x) = \frac{1}{Z(x)} \exp \sum_{k=1}^{K} w_k f_k(y, x) \]  

\[ Z(x) = \sum_{y} \exp \sum_{k=1}^{K} w_k f_k(y, x) \]  

### 3.3.2 Model Training

Before applying the CRF model for anomaly detection, it is necessary to estimate the model's parameters \( w \). Maximum Likelihood Estimation (ML) is the most commonly used method for parameter estimation of CRF models [13]. In order to simplify the operation, a log-linear model is generally selected for parameter estimation. The likelihood function is

\[ L(w) = \sum_{x,y} \tilde{P}(x,y) \log P_w(y^{(i)}|x^{(i)}) \]  

Bring formulas (6) and (7) into \( L(w) \), then

\[ L(w) = \sum_{x,y} \tilde{P}(x,y) \sum_{k=1}^{K} w_k f_k(y, x) - \sum_{x,y} \tilde{P}(x,y) \log Z(x) \]  

\( g(w) \) is the function which \( L(w) \) seeks partial guidance on parameter \( w_k \)

\[ g(w) = \sum_{x,y} \tilde{P}(x,y) P_w(y|x) f(x,y) - \sum_{x,y} \tilde{P}(x,y) f(x,y) \]  

This paper uses the L-BFGS [12] algorithm to learn the CRF model. Firstly, the collected feature parameters are dimensioned and discretized, and the feature parameters are marked to form \( K \) training sequences \( \{X^k|k=1, 2, \ldots, K\} \) with a certain time series. Then, a CRF model is constructed for each training sequence \( X^k \), and \( g(w) \) of each training sequence is calculated to be brought into the L-BFGS algorithm to obtain an optimal \( w \). If the termination condition of the algorithm is not satisfied or the maximum number of iterations is not exceeded, the sequence \( X^{k+1} \) will be trained until the iteration termination condition is satisfied to output the optimal \( w \). Through the above learning process, the CRF anomaly detection model can be finally obtained.

### 3.3.3 Model Anomaly Detection

Before the anomaly detection, the sequence of observations needs to be marked, that is, the prediction problem of conditional random field. In this paper, the well-known Viterbi algorithm [14] is used to obtain the labeled value \( y_1, y_2, \ldots, y_n \) of the observed sequence.

In this paper, the microservice fault matrix is constructed by using the value of the observation sequence \( X(x_1, x_2, \ldots, x_n) \) and its corresponding label value \( Y(y_1, y_2, \ldots, y_n) \). This matrix describes the causal relationship between the type of faults generated during the operation of the microservice system and the fault phenomenon. For ease of understanding, Figure 3 gives an example of a 5×8 microservice fault matrix. The vertical axis of the matrix represents the phenomenon of failure, represented by the \( X \) sequence; The horizontal axis of the matrix represents the type of fault, represented by the sequence \( Y \); the dot in the matrix represents the correspondence between the fault
type and the fault phenomenon, represented by the sequence $XY$; The lower part of the matrix represents the cause of the fault that caused the relevant fault type, and the causal relationship can be derived from the microservice anomaly classification tree. In the constructed microservice fault matrix, each fault cause represents a set $f$, and the elements in the set are $x, y$, that is, the dots in the matrix. In the microservice anomaly detection, the microservice anomaly detection can be realized by using the observation sequence and the corresponding annotation sequence to form a new sequence, and then performing the retrieval in the fault cause. The specific anomaly detection is shown in Algorithm 1.

![Figure 3. Microservice fault matrix.](image)

**Algorithm 1: CRF Model anomaly detection algorithm**

**Input**: CRF model $P$, observation sequence $X(x_1, x_2, \cdots, x_n)$ and abnormal cause set $F(f_1, f_2, \cdots, f_n)$

**Output**: Abnormal sequence set $C$

1. While(not the last $x_n$) do
2. Use *Viterbi* and $P$ get $Y(y_1, y_2, \cdots, y_n)$, calculate $X,Y(x_1, y_1, x_2, y_2, \cdots, x_n, y_n)$
3. If($x, y \in f_i$) then
4. Set $C \cup f_i$
5. End if
6. End do
4. EXPERIMENT

The computer used in this experiment is Intel(R) Core(TM) i5-2520M CPU@2.50GHz with 8GB of memory; the operating system used is Centos 7.4 (64-bit). The experiment is based on an e-commerce website Sock-shop[15] that simulates the microservice. By deploying its service instance on the open source system Kubernetes, and then using the load generator stress[16], the Linux system network delay tool and some docker commands to specifically create faults. We use the open source system monitoring tool Prometheus to collect system parameters such as CPU utilization and memory utilization, and store them in the Redis cache, and then reduce the dimension and obtain the observation sequence. The experiment extracts 50, 200 and 500 observation sequences respectively, and labels the corresponding observation sequences manually. Then we can use the console and auxiliary tools to analyze the cause of the anomaly and construct a microservice fault matrix. After the model training is completed, in order to test the validity of the model, 500, 700, 900, and 1000 observation sequences are selected for testing.

In order to evaluate the performance of the model, this paper compares with the method proposed in [10], and selects two commonly used evaluation indicators: precision and recall. The formula is defined as follows:

\[
\text{precision} = \frac{tp}{tp + fp}, \quad \text{recall} = \frac{tp}{tp + fn}
\]

(11)

Where: \(tp\) is the detected abnormal data, \(fp\) is the detected normal data, \(fn\) is the undetected abnormal data, \(tn\) is the undetected normal data.

The results of the experimental comparison are shown in Figure 4 and Figure 5:

![Figure 4. Precision rate curve.](image)

![Figure 5. Recall rate curve.](image)

It can be seen from Fig. 4 that when the detection sequence is 500. The precision of the anomaly detection between the HMM model and the CRF model is not much different. As the number of detection sequences increases, the accuracy of the CRF model anomaly detection is gradually increased compared with the HMM, and eventually becomes flat at around 900.
It can be seen from Fig. 5 that the recall rate of the CRF model anomaly detection is better than that of the HMM. As the number of detection sequences increases, the difference between the two is widened. Overall, the CRF model is superior to the HMM model.

5. CONCLUSION
The traditional microservice anomaly detection process is too complicated, low in efficiency, and has few detection factors, and cannot detect anomalies comprehensively. The detection method proposed by this paper constructs the observation sequence by collecting the characteristic parameters of the system firstly, and then labels the observation sequence. At last, after the process of combining the anomaly classification tree with the model learning result, we can use the optimal sequence predicted by the model to search in the abnormal cause set, and then find the corresponding exception. This method has a finer granularity of detection and does not invade the system source code. The experimental results show that the proposed method has higher accuracy than the hidden Markov model.

ACKNOWLEDGMENTS
This work is supported by the National Natural Science Foundation of China (Grant No. 51479021, 51679025) and the Fundamental Liaoning Provincial Natural Science Foundation of China(Grant No.20170520196).

REFERENCES
[1] Zhang J, Huang X F, Li C Y. Design and Implementation of Microservice Framework[J]. Computer Systems& Applications, 2017, 26(06): 259-262.
[2] Fowler M, Lewis J. Microservices. http://martinfowler.com/articles/microservices.html.[2014-03-25].
[3] Wang ZY, Wang T, Zhang WB, Chen NJ, Zuo C. Fault diagnosis for microservices with execution trace monitoring. Ruan Jian Xue Bao/Journal of Software, 2017,28(6):1435-1454 (in Chinese). http://www.jos.org.cn/1000-9825/5223.htm
[4] Xu J, Chen P, Yang L, et al. [IEEE 2017 IEEE 14th International Conference on e-Business Engineering (ICEBE) - Shanghai (2017.11.4-2017.11.6)] 2017 IEEE 14th International Conference on e-Business Engineering (ICEBE) - LogDC: Problem Diagnosis for Declaratively-Deployed Cloud Applications with Log[C]// IEEE International Conference on E-business Engineering. IEEE Computer Society, 2017:282-287.
[5] Chen P, Qi Y, Zheng P, et al. CauseInfer: Automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems[C]. IEEE INFOCOM. IEEE, 2014:1887-1895.
[6] Jia Z C, Xing X. Web Service Diagnosis Based on Bayesian and Multiple Fault Reasoning[J]. Computer Science, 2014, 41(06): 225-230.
[7] Bruning S, Weissleder S, Malek M. A Fault Taxonomy for Service-Oriented Architecture[C]// IEEE High Assurance Systems Engineering Symposium. IEEE, 2007.
[8] Liu L, Xiao XH, Fang L, Xu F. Classification Method of Web Service Faults[J]. Computer Systems& Applications, 2010, 19(08): 258-263.
[9] Zhang JF, Zhang XG, Wang H, et al. Exception Classification and Handling Mechanism of CA-CCML Web Service Composition Language[C]// International Conference on Computer Science & Applications. IEEE, 2017.
[10] Wu J, Zeng WR, Chen HL, Tang XF. Approach of measuring and predicting software system state based on hidden Markov model. Ruan Jian Xue Bao/Journal of Software, 2016,27(12):3208-3222 (in Chinese). http://www.jos.org.cn/1000-9825/5014.htm
[11] Liu S, Yang Y J, Chang D X. A method for detecting abnormal protocol behavior based on conditional random field[J]. Application Research of Computers, 2016, 33(06): 1867-1870+1876.
[12] Li Hang. Statistical Learning Method[M]. Beijing: Tsinghua University Press, 2012.3. [ISBN 978-7-302-27595-4].
[13] Liu W. Research on abnormal behavior detection method based on conditional random field model[D]. University of Science and Technology of China, 2014.
[14] Stratonovich R L . Conditional Markov Processes[J]. Theory of Probability & Its Applications, 2006, 5(2):172 ~ 195.
[15] https://microservices-demo.github.io/
[16] https://people.seas.harvard.edu/~apw/stress/