doi :10.3969/j.issn.1671-1122.2023.03.008

一种面向微服务的多维度根因定位算法

施园 [1,2]，李杨 [1,2]，詹孟奇 [1,2]

（1. 中国科学院信息工程研究所，北京 100093；2. 中国科学院大学网络安全学院，北京 100049）

摘　要：伴随着 Docker 等虚拟化容器技术的逐渐成熟，因其可扩展性、灵活性等特点与微服务架构完美契合，工业界逐渐将微服务架构应用部署在基于容器的云环境下，并用 Kubernetes 等容器编排工具来管理应用的全生命周期。在这样复杂的微服务架构下，如何使用人工智能技术高效发现异常并且定位根因成为重中之重。首先，文章总结了在微服务系统环境下进行异常检测和根因定位所面临的主要挑战和现有的关键技术；然后，针对现有技术异常检测覆盖范围不全面的问题，文章提出了一种基于无监督学习的多维度的异常检测方法，在调用链 Trace 数据的基础上结合服务和机器资源利用数据进行综合分析，确保能够检测出服务响应时间异常的同时，也能够识别服务资源利用异常和环境异常；最后，在异常已知的情况下，为了减少根因定位时间，拓展定位范围和缩小粒度，文章提出了一种轻量的基于异常传播子图的方法，将服务接口和机器节点两种维度的数据统一到异常传播子图中进行根因定位。实验表明，文章所提方法与已有方法相比，定位时间更短，不仅拓宽了根因定位场景，而且准确率也有明显提升。

关键词：容器；微服务；Kubernetes；异常检测；根因定位

中图分类号：TP309　**文献标志码：**A　**文章编号：**1671-1122（2023）03-0073-11

中文引用格式：施园，李杨，詹孟奇 . 一种面向微服务的多维度根因定位算法 [J]. 信息网络安全，2023，23（3）：73-83.

英文引用格式：SHI Yuan, LI Yang, ZHAN Mengqi. A Multi-Dimensional Root Cause Localization Algorithm for Microservices[J]. Netinfo Security, 2023, 23(3): 73-83.

A Multi-Dimensional Root Cause Localization Algorithm for Microservices

SHI Yuan[1,2], LI Yang[1,2], ZHAN Mengqi[1,2]

(1. *Institute of Information Engineering*, *Chinese Academy of Sciences*, *Beijing* 100093, *China*; 2. *School of Cyber Security*, *University of Chinese Academy of Sciences*, *Beijing* 100049, *China*)

Abstract: With the gradual maturity of virtualized container technologies such as Docker, because of its scalability, flexibility and other characteristics that perfectly fit the microservice architecture, the industry gradually deploys microservice architecture applications in container-based cloud environments, and use container orchestration tools

收稿日期：2022-11-13

基金项目：国家重点研发计划 [2019YFB1005200,2019YFA1005201]

作者简介：施园（1996—），男，四川，硕士研究生，主要研究方向为智能运维；李杨（1980—），女，北京，副研究员，博士，主要研究方向为 5G 安全、大数据处理与分析、智能运维；詹孟奇（1996—），男，四川，博士研究生，主要研究方向为网络协议异常检测。

通信作者：李杨 liyang@iie.ac.cn

such as Kubernetes to manage the full life cycle of the application. Under such a complex microservice architecture, how to use artificial intelligence technology to efficiently find abnormalities and locate the root cause becomes the top priority. First, the article summarized the main challenges and existing key technologies for anomaly detection and root cause localization in the context of microservice systems. Then, aiming at the problem that the coverage of existing anomaly detection was not comprehensive, we proposed a multi-dimensional anomaly detection method based on unsupervised learning, it combined service and machine resource utilization data for comprehensive analysis on the basis of call chain Trace data to ensure that service response time anomalies can be detected, and service resource utilization anomalies and environmental anomalies can also be identified. Finally, in the case of known anomalies, in order to reduce the root cause localization time, expand the localization range and reduce the granularity, we proposed a lightweight anomaly propagation subgraph-based method. It unified the data of the two dimensions of service interface and machine node into the anomaly propagation subgraph for root cause localization. The experiments results show that proposed method has shorter localization time compared with the existing methods, and not only broadens the root cause localization scenario, but also has a significant improvement in accuracy.

Key words: container; microservices; Kubernetes; abnormal detection; root cause localization

0 引言

随着计算机技术的快速发展，软件开发需求更加灵活多变，在同一软件上进行功能拓展和变更变得更频繁，如何提升软件开发的敏捷性和交付的持续性成为企业面临的难题。目前，由分布式架构演变而来的微服务架构[1]将单个应用拆分成若干个可独立部署的微服务，微服务之间通过轻量级的通信协议协同。不同的微服务围绕自己独立的业务进行开发，可采用多样化的编程语言，选取不同的后台存储技术。这样的特征便于软件的敏捷开发和持续交付，为传统软件架构中面临的问题提出了更好的解决方案。

现如今，Docker[2]等虚拟化容器技术逐渐成熟，其高性能、可扩展性和灵活性等特点与微服务架构完美契合，越来越多的微服务架构应用被部署在基于容器的云环境中，这样每个服务都能获得独立的运行环境、计算和存储资源。与此同时，工业界还采用Kubernetes[3]等容器编排工具对容器的进行管理，它能提供应用创建、应用部署、扩容缩容和应用更新等功能，为管理微服务系统的全生命周期提供了保证。

在庞大复杂的微服务系统中，需要关注的监控指标和数据纷繁错杂，传统的基于领域知识的运维方式已不能满足工业界的要求，越来越多的企业将人工智能和大数据等技术运用于IT运维领域[4]，以弥补自动化运维的不足。然而，如何智能高效地在微服务架构下进行异常检测和根因定位依然面临着许多挑战，通常来说，可以总结为以下3个方面：1）微服务数量巨大，异常检测和定位成本高、速度慢；2）服务之间存在复杂的调用关系，定位到真正根因困难；3）根因定位往往在系统故障发生后，实时性差。

针对以上挑战，出现了各类基于有监督和无监督的异常检测算法，在异常检测方面，它们要么基于集群机器指标进行分析，要么基于微服务的Trace数据进行分析，要么基于系统的日志进行分析。但是没有一种模型能够有效地结合这几种数据进行分析并具有可解释性。在根因定位方面，已存在的根因定位算法要么基于多维度数据进行挖掘，要么需要实时维护全面的故障传播图，这都需要同时分析大量的数据，在实时性上有所欠缺。

因此，本文在充分研究以上算法的基础上，提出了结合机器和服务两个维度进行分析的异常检测算法和基于故障传播图子图的根因定位方法，这为在微服务云环境下，更轻量、高效地进行异常检测和根因定

位提供了一种解决方案。

1 相关研究

在容器环境的微服务体系下，异常检测和故障根因定位大致分为基于日志、基于微服务调用链Trace数据和基于系统监控指标Metric 3个方向。

异常检测主要针对预定义指标的时间序列进行检测，以识别系统运行中的异常行为。通常，关于时间序列[5]的异常标签和先验知识很难获取，针对某个指标，本文根据它的时间序列与历史数据的偏离情况来判定异常，例如，出现数据突降或抖动，说明此指标对应的组件出现了问题。算法的设计对实时性要求很高，时间序列的异常表现形式也多种多样。总的来说，异常检测算法研究的根本目的是更多地、更准确地捕获时间序列中的异常，减少重复的、错误的告警。

故障根因分析旨在从一组异常告警中推测出IT系统的根源故障[6]。在云环境下的微服务系统中，系统组件之间的关系错综复杂，一旦局部组件出现了异常行为，经过故障传播，短时间内可形成大量并发告警。因此，为了识别真正重要的告警，必须在海量的信息中，排除大量无意义的告警，许多企业也因此引入机器学习等技术来实现更高效、更智能化的故障排除模式，使得故障发生时，能够尽可能高效快速地定位根因，减少企业损失。

1.1 异常检测方法研究现状

一般来说，微服务下异常检测主要分为基于监督学习和无监督学习的两种方法。

1.1.1 基于监督学习的异常检测方法

基于监督学习的异常检测方法主要利用带标签的样本数据来训练模型，并且进行类别预测。例如，GUO[7]等人基于监督学习的深度学习提出的一种异常检测模型Translog，它首先通过日志解析的方式将所有原始日志数据[8]解析为相应的日志模板，然后这些日志模板再通过一个编码器[9]将日志模板中的语义信息充分捕捉，最后测试日志通过解码器[10]输出结果0或者1，其中，0代表日志数据是正常的，1代表是异常的，这个过程本质上是一个二分类工作。此外，TIMCENKO[11]等人利用机器学习中的集成分类器[12]对网络入侵现象进行检测也是利用监督学习的方法。然而，在系统运行环境下，异常数据很难获取，即使获取到了，其真实性也很难得到保证，其类别分布也存在不均衡的问题。因此，基于监督学习对微服务系统进行异常检测的方法具有较大的局限性。

1.1.2 基于无监督学习的异常检测方法

基于无监督学习的方式进行异常检测主要是通过对数据进行计算分析，识别出数据集合中相对孤立的点，把这些孤立点看作异常点。在微服务架构下，主要分为基于服务调用链[13]的异常检测和基于指标量化[14]的异常检测两种方法。

基于服务调用链的异常检测将服务调用链和其他服务相关的关键性能指标相结合，构建向量，使用基于非监督学习的各类模型对其进行训练并检测，例如，LIU[15]等人使用深度贝叶斯网络训练模型，挖掘潜在的异常路径。

基于指标量化的异常检测主要利用参数估计和非参数估计的方法量化各类指标，以评估系统中节点的运行情况。其中，基于参数估计的方法典型的有高斯模型和混合参数分布模型[16]等，这类模型预先假设数据服从相应分布并利用样本的特征值训练模型，在检测时将异常分数定义为数据实例到估计平均值的距离。如果模型假设的数据分布和实际有差别，就不一定能够得到准确的结果，而且在复杂场景下，一些数据分布很难用简单的函数进行描述。因此，基于参数估计的方法在指标类型复杂的场景下有很大局限性。

基于非参数估计的异常检测方法不需要对数据分布进行假设，不依赖于先验知识。例如，AHMED[17]等人提出了基于核密度估计的异常检测方法去估计出数据的概率密度函数，并将偏离分布的数据实例标记为异常。该方法能有效检测出时间序列中存在的异常，但相对更耗时，不适用于对检测时间有要求的场景。

1.2 根因定位方法研究现状

在微服务体系下，根因定位算法主要从多维度数据挖掘和故障传播图两个方向进行研究。

1.2.1 基于多维度数据挖掘

这类方法利用系统运行过程中产生的大量维度数据进行分析，当某一个关键指标发生异常的时候，利用机器学习算法快速准确定位到是哪个交叉维度的细粒度指标异常所导致的，以便尽快修复止损。这类算法主要分为关联规则挖掘和设定一个判别根因的目标函数进行启发式搜索两类。以关联规则挖掘为例，AHMED[18]等人和LIN[19]等人提出的方法都是首先对所有属性组合进行异常检测，然后用Apriori算法去搜索超过阈值的属性组合，信息也和服务的状态密切相关，最后根据置信度筛选出和异常类所关联的属性组合。与此类似，SUN[20]等人将关联规则挖掘看作是挖掘不同类中分布差别较大的属性集合并取得较好的效果。因此基于关联规则挖掘的方法在参数合适的情况下，可以取得非常好的效果，但是随着数据集和故障案例的变化，参数需要时刻修正，效果不稳定。

基于启发式的搜索方法需要定义需要关注的根因，并据此定义一个目标函数，然后在整个搜索空间中搜索使得目标函数最大化的属性组合作为根因。由于搜索空间往往比较大，所以需要利用启发式方法或剪枝方法缩小搜索空间。例如，SUN[20]等人考虑了有多个根因同时作用的情况，并且使用蒙特卡洛搜索算法来进行启发式搜索，GU[21]等人对目标函数做了一些改进，同时采用启发式的策略来加速搜索，避免性能损失。对于根因的属性数量比较大，指标变化相比整体不够显著的时候，该算法效果相对更稳健。总的来说，基于启发式的搜索方法比盲目型的搜索方法更高效，但其算法效果和启发式规则的设计紧密相关，在微服务系统实际场景下，很难设计出这样效果稳定的启发式规则。

1.2.2 基于故障传播图

故障传播图描述了系统异常的时候节点之间的关联信息，在对图中节点进行重要性排序之前，本文需要先维护一个异常调用图模型并定义异常传播概率。一般来说，基于故障传播图的根因定位主要利用PageRank算法[22]和随机游走算法[23]。一般云原生场景下的故障传播图是有向无环图，因此PageRank算法得到的节点的重要程度排名可以作为各个节点对该异常的影响度排名。随机游走算法则是利用随机游走过程捕获不同系统组件之间可能的异常传播路径，进而根据节点访问频率定位根因。其中，随机游走过程模仿人员的手动分析行为[24]，并形成了马尔可夫链[25]。此类方法以服务为导向，由系统终端异常触发，适合于故障发生后的根因分析场景，但是局限于定位机器根因水平，不适合系统的实时定位场景。

2 面向微服务的多维度根因定位算法实现

在微服务环境下进行根因定位，需要先检测出系统异常，但是由于云环境下微服务系统数据量大，标签少，异常情况较少且异常标签很难判定，采用监督学习的方法很难取得好的效果。因此，本文采用非监督学习的方法学习正常数据的特征，又考虑到微服务系统中服务之间调用关系复杂，单独依靠日志[26,27]和机器指标信息[28,29]不能够帮助本文有效地发掘服务之间的异常，而服务之间的调用链Trace[30]数据详细地记录了服务之间的调用关系以及响应时间。因此，本文的异常检测在微服务系统调用链Trace数据的基础上进行。又考虑到微服务系统环境复杂，单一的指标如响应时间不能够完全代表服务的状态，多备份云环境下，服务的资源使用量和服务所在的云主机环境信息两者与服务的状态密切相关。

综上所述，本文提出一种准确轻量、灵活、可扩展的并且基于微服务调用链的异常检测框架。该框架在微服务系统调用Trace数据的基础上，结合微服务的资源使用量以及服务所在环境的信息摘要统一进行学习，无需构建任何关系图[31]，就能够直接对异常数据进行检测。如图1所示，本文定期采集Traces数据以及服务相关的环境信息构建向量，使用无监督学习的方法进行线下训练，生成基于多指标的异常检测模型。

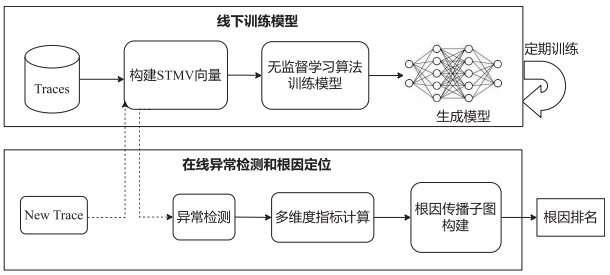


图 1 多维度根因定位算法框架

传统的基于故障传播图的根因定位方法如文献[31]所描述，需要实时维护一个庞大的基于服务节点的图模型，其数据处理量大，模型效果极度依赖图模型的构建，因此不适用于变更频繁的微服务系统。本文在多维度指标异常检测模型的基础上，当异常时间点已知的情况下，利用一段时间内的微服务Trace数据、服务所使用的CPU、Memory数据以及云主机的CPU、Memory、Network数据构建将服务接口和云主机统一的异常传播子图。如图1所示，当一条Trace数据和Metric服务构建的向量被检测为异常时，本框架将利用一段时间内环境中的机器指标数据和服务接口数据构建异常传播子图，并进行异常根因定位，排名找出最可能的根因。

3 异常检测算法模型设计

为了更好地学习服务特征，本文提出构建基于服务的结合调用链和机器指标的异常检测向量（Service Trace Metric Vector，STMV），在此基础上，利用改进的VAE模型[32]训练数据，检测异常。本模型包含STMV向量构建、模型训练和异常检测三个模块。

3.1 STMV 向量构建

3.1.1 构建 Trace 和响应时间相结合的向量

LIU[15]等人将微服务调用链和响应时间有效地结合，以用户uuid为单位构建服务调用链向量（Service Trace Vector，STV）训练模型对异常Trace数据进行检测，取得了较好的效果，如图2所示。在STV中，每个维度的键代表调用当前服务接口时，该Trace数据已经经历的服务调用链，值是调用当前服务时的响应时间。这样构建向量的方法，在微服务数量较小且调用关系简单的时候，维度可控。但在微服务数量较大，服务间调用关系复杂的情况下，向量的维度和模型线下训练的时间将以指数形式增加，影响模型的效果。

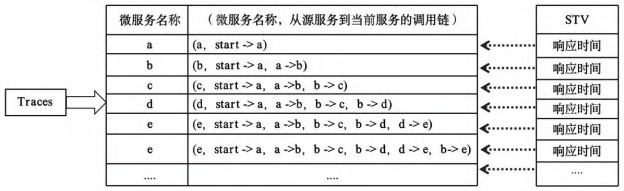


图 2 STV 向量模型

因此，本文调整服务调用链和响应时间的对应关系，以单个Trace数据为单位，以服务接口为粒度构建向量，如图3所示。

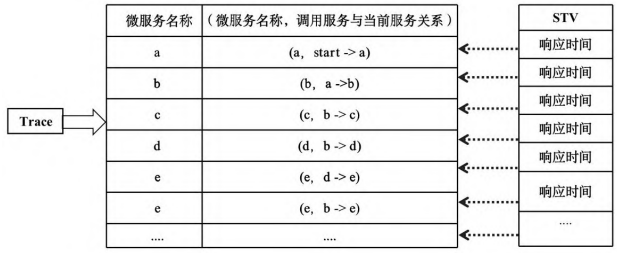


图 3 单个 Trace 向量构建

由于本模型是对每一条Trace进行异常检测，微服务的接口数量固定，因此向量的维度不会随微服务数量的增加或调用关系的变化而变化。

如表1和图4所示，对每条Trace数据构建向量，*T*1、*T*2拥有相同的结构，图4 a）为正常trace数据，正常调用过程分别为①、②、③、④、⑤、⑥，其中调用过程⑤为异步调用。图4 b）表示由于服务b异步调用服务e错误的异常Trace数据，图4 c）表示由于服务b响应超时的异常Trace数据。在Trace向量中，对没有使用维度的响应时间填充0，如表1中*T*3列一样，*Dimension ID*为5和6的维度响应时间设置为0。

表 1 服务响应超时异常和调用异常

Dimension ID	*Callpath*	*T*1	*T*2	*T*3	*Anomalous*
1	(a, start->a)	222	1302	1102	Yes
2	(b,a->b)	209	1138	1305	Yes
3	(c,b->c)	4	9	6	No
4	(d,b->d)	44	36	40	No
5	(e,d->e)	30	32	0	No
6	(e,b->e)	67	980	0	Yes

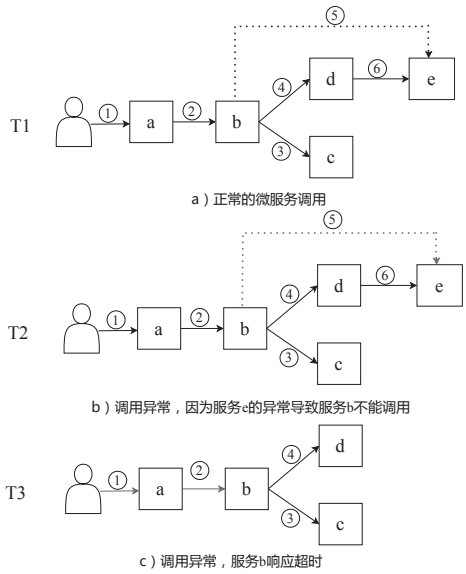


图 4 服务响应超时异常和调用异常图

3.1.2 结合环境信息构建向量 STMV

目前基于Trace数据的异常检测方法没有考虑服务自有资源的使用情况，也没有考虑服务所在环境的因素，因此本文将服务的CPU和Memory资源使用情况作为服务的基本属性，纳入向量构建之中。又考虑到微服务系统在分布式的部署环境下，存在高备份情况，每个服务都可能备份在多个节点之上，其中节点环境的变化能够侧面反映该微服务的状态，因此节点环境的变化也是异常检测的重要一环。例如，当微服务所在节点因为非系统原因出现巨大变化时，此时该微服务的响应时间和资源使用情况均正常，但此状态应该表现为异常。

在微服务系统环境中，节点数量巨大，每个微服务备份一个或者多个节点，每个节点状态的衡量指标数量大，类型多，且存在相关性。因此，本文利用主成分分析法[33]（Principal Component Analysis，PCA）针对每个服务所在主机的指标，在集群下取其主成分纳入构建向量中，若每个服务存在三备份，PCA序列就是每个服务在3个节点下CPU、Memory和Network构成的序列摘要，如图5所示。

其中，微服务 a 部署在云主机节点 1、节点 2 和节点 3 上，则其微服务环境摘要就是在这 3 个节点下

微服务环境摘要	所在云主机节点	PCA序列
服务a的PCA值	节点1、节点2、节点3	节点1、2、3各自cpu使用量、节点1、2、3各自内存使用量、节点1、2、3各自网络吞吐量
服务b的PCA值	节点2、节点3、节点4	节点2、3、4各自cpu使用量、节点2、3、4各自内存使用量、节点2、3、4各自网络吞吐量
服务c的PCA值	节点4、节点5、节点6	节点4、5、6各自cpu使用量、节点4、5、6各自内存使用量、节点4、5、6各自网络吞吐量
服务d的PCA值	节点6、节点7、节点8	节点6、7、8各自cpu使用量、节点6、7、8各自内存使用量、节点6、7、8各自网络吞吐量
服务e的PCA值	节点8、节点9、节点10	节点8、9、10各自cpu使用量、节点8、9、10各自内存使用量、节点8、9、10各自网络吞吐量

图 5 微服务环境信息的 PCA 摘要

CPU、Memory 和 Network 构成的 PCA 序列摘要值。得到所有微服务的环境摘要后，为了使向量具有可解释性，并且尽可能保证特征的关联性，本文构建 STMV 如图 6 所示。其中，cpu usage 和 memory usage 分别表示该服务总的 CPU 和 Memory 使用量，PCA abstract 表示该服务所在机器环境各项指标与总体环境指标的 PCA 分析值，即图 5 所示微服务环境摘要值。

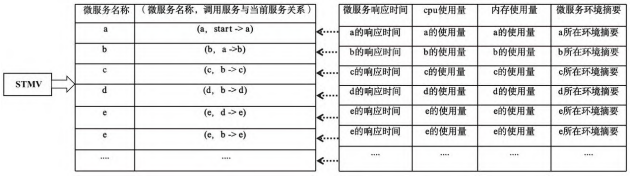


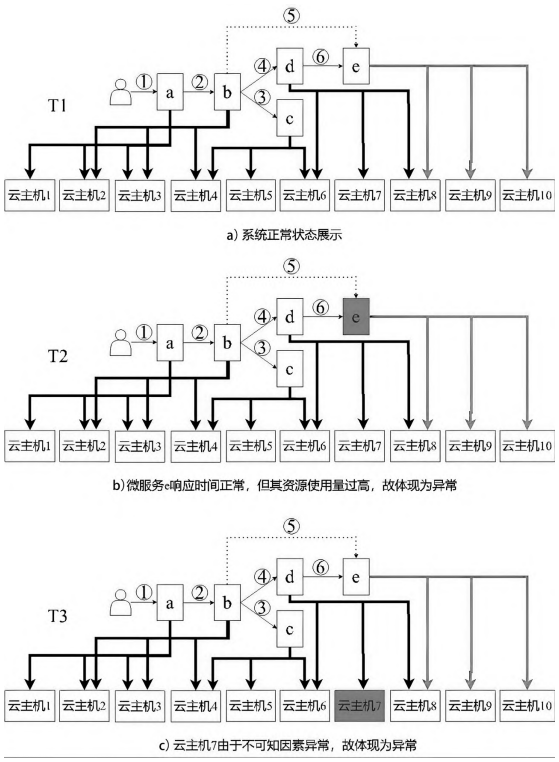
图 6 STMV 向量构建

如图7所示，每一个服务都处于3个备份的状态，如微服务a分别部署于云主机1、云主机2和云主机3三个节点，图7 a）是系统正常的状态，服务的调用关系依次为①~⑥顺序，其中⑤号调用为异步调用过程。

图7 b）是当服务e本身CPU资源使用量或者内存使用量异常导致服务异常，对应表2的STMV ID 为6的场景，图7 c）是云主机7由于不可知的原因导致资源使用异常的情况，此状态也应识别为异常，对应表2中STMV ID 为7的场景。结合前文提及表2中对应的STMV ID为4和5分别表示的服务响应超时异常和服务调用异常，则本文所检测的异常覆盖了微服务和云主机两种场景。

3.2 模型训练

本文采用VAE训练模型，VAE训练模型作为一个生成模型，把大量真实样本通过编码器网络变换成一个理想的数据分布，该数据分布再传递给一个解码器网络，最后得到大量生成样本，生成样本与真实样本足够接近时，则训练出自编码器模型。为了进一步



a) 系统正常状态展示

b) 微服务e响应时间正常，但其资源使用量过高，故体现为异常

c) 云主机7由于不可知因素异常，故体现为异常

图 7 STMV 所针对的微服务负载异常和节点异常场景

表 2 STMV 真实示例

向量状态	STMV ID	a 的响应时间 /ms	b 的响应时间 /ms	a 的 cpu 使用量 /core	b 的 cpu 使用量 / core	a 的内存使用量 / byte	b 内存使用量 /byte	a 的摘要	b 的摘要
正常	1	222	209	0.207	0.219	2765402112	3866624	0.8279	1.7778
正常	2	198	203	0.244	0.239	2768973824	3903488	0.8465	1.8079
正常	3	212	204	0.232	0.219	2780045312	4788224	0.8562	1.7898
异常	4	1302	1138	0.209	0.282	2680045312	4687824	0.8892	1.7989
异常	5	1102	1305	0.392	0.246	2700045312	4695824	0.8578	1.7789
异常	6	232	198	2.827	0.267	2710045435	4995912	1.012	1.8789
异常	7	221	178	0.401	0.298	2980045345	3803483	1000.2	28.2

精确解码器的输入特征，本文利用流传输的方法改进VAE模型，首先输入向量$\boldsymbol{x}$进入深度贝叶斯网络构建的编码器网络中，获取其隐藏特征，然后通过这些隐藏特征可以得到均值$\mu_{z(0)}$和标准差$\sigma_{z(0)}$，$z^{(0)}$从均值和标准差的高斯分布中采样，最后通过长度为K的向后传输流，得到$z^{(K)}$如公式（1）所示。

$$z^{(K)}=\left(f^{(K)},\cdots,f^{(1)}\right)\left(z^{(0)}\right) \quad (1)$$

其中，$z^{(K)}$作为潜在特征变量，进入解码器生成网络中，通过隐藏层获取其隐藏特征，得到均值$\mu_{x(0)}$和标准差$\sigma_{x(0)}$，重构的向量$\boldsymbol{x}$从此均值和标准差构建的高斯分布中采样获得。本模型使用KINGMA[34]等人提出的SGVB方法训练模型，在训练过程中利用输入向量和生成向量的模型参数。因为本文的异常检测是无监督的，训练数据中默认有少量异常数据，所以使用随机梯度下降的方法训练模型能够捕捉数据分布最重要的特征。

3.3 实验效果

本文所提模型的的异常检测通过相似度比较进行判断，其使用核密度估计（Kernel Density Estimation，KDE）的方法去学习正常STMV的分布，若一个STMV的值明显大于或者小于正常分布的值，则判定为异常，其他情况判定为正常。

为了验证本文所提算法，将其与目前较为广泛使用的算法（如TraceAnomaly、OmniAnomaly等）模型进行比较。本文搭建开源微服务测试平台TrainTicket，其中包含41个微服务。在数据采集过程中中，本文使用开源链路追踪系统Jaeger去采集微服务系统的Trace数据，使用开源监控平台Prometheus监控测试平台并采集微服务关联的机器指标等信息，使用阿里巴巴开源混沌测试工具ChaosBlade控制系统环境。

本文收集一段时间内的系统日常数据为训练数据，从两方面验证该方法的有效性：1）评估系统对异常响应时间和异常调用路径的检测效果；2）评估系统对微服务CPU、Memory异常以及系统环境异常的检测效果。

对比多个模型，结果如表3所示，本文的方法在针对微服务响应时间和调用路径的异常检测上，效果和最新的TraceAnomaly相似，准确率和召回率在0.9左右，在微服务负载异常检测和微服务系统环境异常检测上，效果明显高于已有方法。

实验结果表明，本文所提异常检测算法模型具有鲁棒性，无论是复杂大型微服务系统还是小型微服务系统，其训练时间和成本变化不大，并且该算法还提高了微服务异常检测的全面性，能够检测出服务除调用链和响应时间以外环境的异常。

表 3 算法对比结果

算法名称	响应时间异常		调用路径异常		微服务负载异常		微服务系统环境异常	
	准确率	召回率	准确率	召回率	准确率	召回率	准确率	召回率
Hard-coded Rule 算法	0.89	0.79	N/A	N/A	N/A	N/A	N/A	N/A
Multimodal LSTM 算法	0.62	0.95	N/A	0.93	0.53	0.67	0.69	0.89
Auto-Encoding Variational Bayes 算法	0.16	0.52	0.17	0.98	0.64	0.57	0.71	0.84
Omni-Anomaly 算法	0.45	0.49	0.46	0.94	0.6	0.94	0.65	0.91
Trace-Anomaly 算法	0.98	0.97	N/A	0.89	N/A	N/A	N/A	N/A
本文算法	0.97	0.98	N/A	0.91	0.91	0.98	0.93	0.99

4 根因定位算法模型设计

由于单个微服务中存在多个接口，针对不同的请求，则其响应状态有所不同。现在的工作大多以微服务为最小单位进行根因定位，忽略了接口这个最小执行单元状态的重要性，因此本文的根因定位在微服务接口数据的基础上进行。此外，目前广泛使用的基于图模型的算法仅将服务节点考虑为异常传播图的节点，虽然 WU[35] 等人利用容器和云主机的资源使用情况作为判断服务状态的依据，但是在定位时却将云主机排除在外，因此这些模型都不能定位未知的机器水平的异常。本文提出一种可解释的状态量化方法，将服务接口和主机统一到异常传播图中，并利用改进的 Personalized PageRank 算法进行根因排序。

本文在初步筛选出异常节点后，通过自定义的边权赋值方法统一为图的边赋权重值，以便使用随机游走算法调整节点的重要程度。KIM[24] 等人为每个节点设置初始状态值，有效提升了预测图节点重要性的准确度。因此，本文为异常节点定义初始异常分数，使用 Personalized PageRank 算法进行根因排序。根因排序主要包括 3 个流程：异常故障传播子图构建、异常节点筛选和边权赋值和故障根因定位。

4.1 异常故障传播子图构建

当异常发生时，本文认为异常情况存在于当前时刻相关联的服务或者云主机之中，因此，本文根据当前时间段的 Trace 数据解析出服务接口（service:api）之间的关联图。如图 8 所示，接口 s1:api2 调用了接口 s2:api1，那么存在一条从接口 s1:api2 指向接口 s2:api1 的有向边。其中，service:api 之间相互调用，且多个 api 接口同属于一个服务，如图 8 中同一椭圆形内的接口属于同一服务。

由于每个服务存在多备份的情况，即服务会同时运行在多个主机之上，如图 8 中指向主机 h1 的边所示，因为服务 s1 在主机 h1 上有部署，所有存在一条由服务 s1 指向主机 h1 的有向边。在实际构建中，同一服务的每个接口都指向服务所关联的主机。

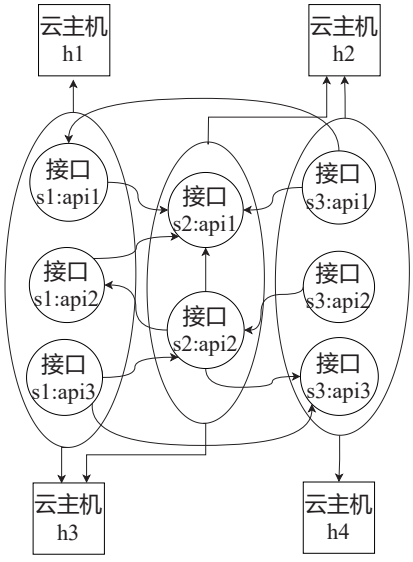


图 8 异常传播子图构建

4.2 异常节点筛选和边权赋值

在完成构建异常传播图后，本文需要根据每个节点的属性，初步筛选出异常的节点。对于每个服务接口，其拥有响应时间、服务 CPU 利用率和 Memory 利用率 3 个属性，其中响应时间数据是利用服务 Trace 数据解析得到，CPU 利用率和 Memory 利用率是该服务的资源使用情况，多个接口共用同一服务的资源使用数据。对于每个主机，本文采集其 CPU 利用率、Memory 利用率和网络吞吐量 3 个属性。当异常发生时，本文利用核密度估计的方法分别提取这 3 个属性的时序数进行评估，如果一项指标明显偏离正常，那么此节点或者链

路为异常。如图9所示，接口s2:ap1对外部请求响应时间超时，则存在s1:api1指向s2:ap1、s1:api2指向s2:ap1、s3:api1指向s2:api1的异常边。对于服务s1而言，若整个服务的资源利用率明显偏离正常值，则整个服务的接口节点都标记为异常，因此存在s3:api1指向s1:api1、s2:api2指向s1:api2的异常边。主机节点h3由于资源使用异常而被标记为异常，因此指向主机h3的节点边均为异常边。

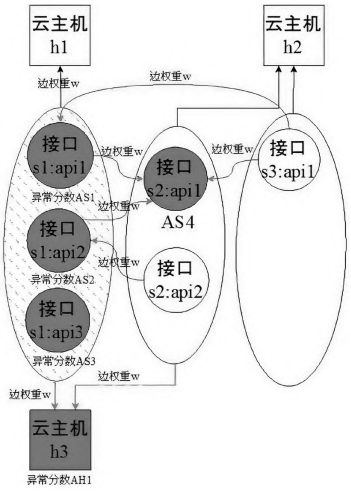


图 9 异常节点筛选和边权赋值

在筛选出异常节点和异常边之后，和异常节点不相关的节点可以默认为正常节点。为了减少计算复杂度，使本文所提模型更轻量，本文在异常传播子图的基础上去除掉没有异常可能性的边和节点。

对于异常的服务之间和异常的服务和主机之间，本文使用0到1之间的常量来表示其边的权重，设置异常置信度，在算法训练中，若这个节点为异常节点，则置信度的值会逐渐增加。在利用异常传播图进行根因定位的过程中，节点所链接的边的数量和权重是评估节点重要程度的两个重要因素，因此在评估异常服务节点和正常服务节点边的权重时，本文利用皮尔森相关系数评价其最大相关系数，如公式（2）所示。

$$w_{ij}=\max_{k:u_k\in U_n(j)}\left(corr\left(u_k,node(i)\right)\right) \tag{2}$$

其中，w_{ij}是异常服务节点i和正常节点j之间的权重，若u_k是服务节点，则其包含了响应时间、服务 CPU 使用量和服务 Memory 使用量。若u_k是主机节点，则其包含了网络吞吐量、CPU 使用量和 Memory 使用量，同时，$node(i)$表示异常节点 i，包含以上 3 个维度的数据。当i和j属于同一服务时，u_k仅包含响应时间。为了总结异常节点的异常分数，节点s_j的异常分数$AS(s_j)$如公式（3）所示。

$$AS(s_j)=\frac{\sum w(s_j)}{n} \tag{3}$$

其中，n为该异常节点边的数量。

4.3 故障根因定位实验效果

基于图中心的Personalized-pageRank算法[36]已在捕获异常传播中具有非常好的效果。在此模型中，个性化PageRank向量（Personalized PageRank Vector，PPV）被视为每个节点的根因分数，为了计算得到PPV，本文定义若$node_i$到$node_j$存在链接，则传播概率矩阵$P_{ij}=\frac{w_{ij}}{\sum_j w_{ij}}$；否则$P_{ij}=0$，最终节点的根因分数$v=(1-c)Pv+cx$，其中$x=AS(s_j)$，$c=0.15$。

本文的实验环境包含4台8核心、16GB的云主机，其中1台主机为Kubernetes Master节点，剩余3台主机为Slave节点，接受服务调度的主机为3台Slave主机，每个服务3个备份。本实验主要从两个方面进行评估：1）针对服务响应时间异常根因进行评估；2）针对服务资源利用异常根因和主机资源利用异常根因进行评估。评估效果如表4所示。

表 4 本文算法和其他根因定位算法比较

Metric	RS 算法	MonitorRank 算法	Microscope 算法	MicroRCA 算法	本文所提算法
服务接口响应超时根因					
PR@1	0.18	0.24	0.71	0.87	0.89
PR@3	0.43	0.57	0.89	0.93	0.95
MAP	0.43	0.57	0.89	0.93	0.95
服务资源利用异常根因					
PR@1	0.21	0.17	0.45	0.76	0.89
PR@3	0.46	0.33	0.67	0.71	0.91
MAP	0.42	0.45	0.65	0.7	0.92
云主机资源利用异常根因					
PR@1	0.1	0.6	0.52	0.78	0.91
PR@3	0.26	0.65	0.56	0.83	0.93
MAP	0.41	0.61	0.6	0.81	0.94

其中PR@1和PR@3分别表示在算法模型结果的

前一和前三目标中包含真正的根因的准确度，MAP表示在对环境中所有节点进行根因测试的平均结果。

如图10所示，本文的方法在针对服务接口粒度上，相对于服务粒度在响应时间根因定位准确率有所提升，另外，本文的方法在云主机异常和服务资源异常根因定位评估上，依然能够保持更高准确率。在时间消耗上，本文的方法也保持了较高稳定性，如图10所示，随着异常传播图中节点数的增加，本文的方法定位耗时缓慢增加，且耗时低于其他算法。

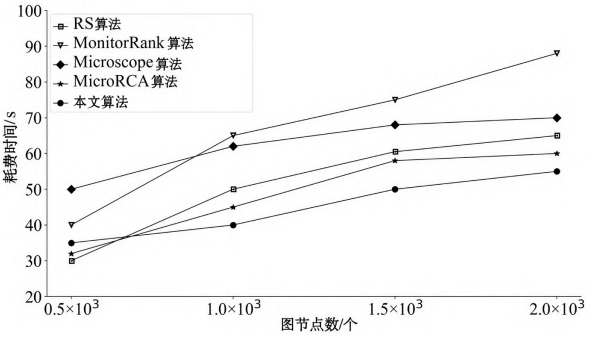


图 10 算法耗时对比

实验证明本文方法具有以下优点：1）轻量高效，根因定位时间少，且无需维护庞大的异常传播图；2）将服务和云主机节点统一进行根因定位，能够准确定位微服务接口粒度和机器粒度的异常根因。

5 结束语

在微服务系统异常检测方面，本文针对目前基于微服务Trace数据进行异常检测的算法没有考虑服务环境异常和服务自身资源利用异常的问题，设计了一种基于多维度指标的异常检测算法。实验证明，本文所图算法在响应时间和调用链异常以及微服务资源利用异常、环境异常上都能保持较高准确率。在微服务系统根因定位方面，针对已有算法需要实时维护庞大的故障传播图且都是针对服务进行定位的问题，本文提出了一种将服务接口和主机节点共同纳入异常传播子图构建并进行根因定位的算法。实验证明，本文的方法在覆盖微服务系统根因定位的3个场景上，准确度均有提升。在不同规模的节点进行根因定位时，本文算法的耗时稳定且更短。

随着智能运维的快速发展，可能会出现将微服务Trace数据、系统日志数据和系统监控数据相结合的综合性异常检测和根因定位算法。在工业界复杂的微服务系统中，为了充分利用已有的经验知识，将监督学习和无监督学习两种方法结合应用也是未来的研究方向。

参考文献：

[1] LI Zhenhao. Development and Impact Analysis of Microservice Architecture[J]. China CIO News, 2017, 1: 154−155.
李贞昊 . 微服务架构的发展与影响分析 [J]. 信息系统工程，2017，1：154−155.
[2] BOETTIGER C. An Introduction to Docker for Reproducible Research, with Examples from the R Environment[J]. ACM SIGOPS Operating Systems Review, 2015, 49(1): 71−79.
[3] BERNSTEIN D. Containers and Cloud: From LXC to Docker to Kubernetes[J]. IEEE Cloud Computing, 2014, 1(3): 81−84.
[4] ZHANG Xuejian, ZHANG Yu, CHUAN Tao, et al. Construction Method of It Operation and Maintenance Data Management System Based on Big Data Technology[J]. Electronic Science and Technology, 2018, 31(4): 84−86.
张雪坚，张榆，钏涛，等 . 基于大数据技术的 IT 运维数据管理系统构建方法 [J]. 电子科技，2018，31（4）: 84−86.
[5] WANG Wei, SHEN Xudong. Research on Anomaly Detection Al−Gorithm of Migration Time Series Based on Instance[J]. Netinfo Security, 2019, 19(3): 11−18.
王伟，沈旭东 . 基于实例的迁移时间序列异常检测算法研究 [J]. 信息网络安全，2019，19（3）: 11−18.
[6] LAN Qing. Application Analysis of Intelligent Operation and Maintenance in Enterprise Mmanagement[J]. Electronic World, 2020(5): 89−90.
兰清 . 智能运维在企业 IT 管理中的应用分析 [J]. 电子世界，2020（5）: 89−90.
[7] GUO Hongcheng, LIN Xingyu, YANG Jian, et al. TransLog: A Unified Transformer−Based Framework for Log Anomaly Detection[EB/OL]. (2022−01−17)[2022−11−10]. https://doi.org/10.48550/arXiv.2201.00016.
[8] HE Shilin, ZHU Jieming, HE Pinjia, et al. Loghub: A Large Collection of System Log Datasets Towards Automated Log Analytics[EB/OL]. (2020−09−14)[2022−11−10]. https://doi.org/10.48550/arXiv.2008.06448.
[9] JACOB D, CHANG Mingwei, KENTON L, et al. BERT: Pre−Training of Deep Bidirectional Transformers for Language Understanding[EB/OL]. (2019−05−24)[2022−11−10]. https://doi.org/10.48550/arXiv.1810.04805.
[10] HALL S. Encoding/Decoding[M]. New York: Culture, Media, Language, 1980.
[11] TIMCENKO V, GALIN S. Ensemble Classifiers for Supervised Anomaly Based Network Intrusion Detection[C]//IEEE. 2017 13th IEEE International Conference on Intelligent Computer Communication and

Processing (ICCP). New York: IEEE, 2017: 13−19.
[12] WANG W, KANNEG D. An Integrated Classifier for Gear System Monitoring[J]. Mechanical Systems and Signal Processing, 2009, 23(4): 1298−1312.
[13] XU Yong, ZHU Yaokang, QIAO Bo, et al. Tracelingo: Trace Representation and Learning for Performance Issue Diagnosis in Cloud Services[C]//IEEE. 2021 IEEE/ACM International Workshop on Cloud Intelligence (Cloudintelligence). New York: IEEE, 2021: 37−40.
[14] ZHANG Shenglin, LIN Xiaofei, SUN Yongqian, et al. Research on Unsupervised KPI Anomaly Detection Based on Deep Learning[J]. Frontiers of Data and Computing, 2020, 2(3): 87−100.
[15] LIU Ping, XU Haowen, OUYANG Qianyu, et al. Unsupervised Detection of Microservice Ttrace Anomalies Through Service−Level Deep Bayesian Networks[C]//IEEE. 2020 IEEE 31st International Symposium on Software Reliability Engineering(ISSRE). New York: IEEE, 2020: 48−58.
[16] CHANDOLA V, BANERJEE A, KUMAR V. Anomaly Detection: A Survey[J]. ACM Computing Surveys, 2009, 41(3): 1−58.
[17] AHMED T. Online Anomaly Detection Using KDE[C]//IEEE. Proceedings of the 28th IEEE conference Global Telecommunications. New York: IEEE, 2009: 1009−1016.
[18] AHMED F, ERMAN J, GE Zihui, et al. Detecting and Localizing End−to−End Performance Degradation for Cellular Data Services Based on TCP Loss Ratio and Round Trip Time[J]. IEEE/ACM Transactions on Networking, 2017, 25(6): 3709−3722.
[19] LIN F, MUZUMDAR K, LAPTEV N P, et al. Fast Dimensional Analysis for Root Cause Investigation in a Large−Scale Service Environment[J]. Proceedings of the ACM on Measurement and Analysis of Computing Systems, 2020, 4(2): 1−23.
[20] SUN Yongqian, ZHAO Youjian, SU Ya, et al. Hotspot: Anomaly Localization for Additive KPIs with Multi−Dimensional Attributes[J]. IEEE Access, 2018, 6: 10909−10923.
[21] GU Jiazhen, LUO Chuan, QIN Si, et al. Efficient Incident Identification from Multi−Dimensional Issue Rports via Meta−Heuristic Search[C]//ACM. Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. New York: ACM, 2020: 292−303.
[22] XING Wenpu, GHORBANI A. Weighted PageRank Algorithm[C]//IEEE. Proceedings Of the Second Annual Conference on Communication Networks and Services Research, . New York: IEEE, 2004: 305−314.
[23] FOUSS F, PIROTTE A, RENDERS J M, et al. Random−Walk Computation of Similarities Between Nodes of a Graph with Application to Collaborative Recommendation[J]. IEEE Transactions on Knowledge and Data Engineering, 2007, 19(3): 355−369.
[24] KIM M, SUMBALY R. Root Cause Detection in a Service−Oriented Architecture[J]. ACM SIGMETRICS Performance Evaluation Review, 2013, 41(1): 93−104.
[25] BRIN S, PAGE L. The Anatomy of a Large−Scale Hypertextual Web Search Engine[J]. Computer Networks and ISDN Systems, 1998, 30(1): 107−117.
[26] TAI Liyuan, TIAN Chunqi, WANG Wei. Anomaly Detection of Large Scale Microservice Architecture Software System Based on Log Parsing[J]. Computer Science and Application, 2019, 9(12): 2266−2276.
[27] JIA Tong, YANG Lin, CHEN Pengfei, et al. LogSed: Anomaly Diagnosis Through Mining Time−Weighted Control Flow Graph in Logs[C]//IEEE. IEEE International Conference on Cloud Computing. New York: IEEE, 2017: 447−455.
[28] GULEMKO A, SCHMIDT F, ACKER A, et al. Ddetecting Anomalous Behavior of Black−Box Services Modeled with Distance−Based Online Clustering[C]//IEEE. 2018 IEEE 11th International Conference on Cloud Computing. New York: IEEE, 2018: 912− 915.
[29] SAMIR A, PAHL C. DLA: Detecting and Localizing Anomalies in Containerized Microservice Architectures Using Markov Models[C]//IEEE. 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud). New York: IEEE, 2019: 205−213.
[30] NEDELKOSKI S, CARDOSO J, KAO O. Anomaly Detection and Classification Using Distributed Tracing and Deep Learning[C]//IEEE. 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). New York: IEEE, 2019: 241−250.
[31] MA Meng, XU Jingmin, WANG Yuan, et al. AutoMAP: Diagnose Your Microservice−Based Web Applications Automatically[C]//ACM. Proceedings of the Web Conference 2020. New York: ACM, 2020: 246−258.
[32] AN J, CHO S. Variational Autoencoder Based Anomaly Detection Using Reconstruction Probability[J]. Special Lecture on IE, 2015, 2(1): 1−18.
[33] SHLENS J. A Tutorial on Principal Component Analysis[EB/OL]. (2014−04−03)[2022−11−10]. https://doi.org/10.48550/arXiv.1404.1100.
[34] KINGMA D P, WELLING M. Auto−Encoding Variational Bayes[EB/OL]. (2014−05−01)[2022−11−10]. https://max.book118. com/html/2021/0321/5314343041003201. shtm.
[35] WU Li, TORDSSON J, ELMROTH E, et al. MicroRCA: Root Cause Localization of Performance Issues in Microservices[C]//IEEE. NOMS 2020−2020 IEEE/IFIP Network Operations and Management Symposium. New York: IEEE, 2020: 1−9.
[36] JEF G, WIDOM J. Scaling Personalized Web Search[C]//ACM. Proceedings of the 12th International Conference on World Wide Web. New York: ACM, 2003: 271−279.