DOI:10.20033/j.1003-7241.(2025)06-0072-05

基于知识图谱的微服务架构故障根因定位方法

刘 劲，汪 岳，李志刚，董志强，郭 浩
（中国海洋石油集团有限公司信息技术中心，北京 100010）

摘 要：为提升微服务架构故障定位效果，提出基于知识图谱技术的微服务架构故障根因定位方法。基于多维关联分析算法构建微服务系统的多维数据分类模型，用于判断是否发生故障；构建微服务架构的连接拓扑图，使用知识图谱的PageRank算法来动态调整拓扑图中各服务组件的权重关系，计算出故障概率最大的服务组件作为故障根因，实现微服务架构故障根因定位。测试结果表明，所提方法定位结果准确率高于0.75，召回率在0.71以上，F1得分均在0.75以上，有效提高了微服务架构故障定位效果，具有较强的实用性和可靠性。

关键词：知识图谱；微服务架构；根因定位；PageRank算法
中图分类号：TP277.3；TP391 文献标识码：A 文章编号：1003-7241(2025)06-0072-05

Root Cause Localization Method of Microservice Architecture Based on Knowledge Graph

LIU Jin, WANG Yue, LI Zhi-gang, DONG Zhi-qiang, GUO Hao
(CNOOC Information Technology Center, Beijing 100010, China)

Abstract: In order to improve the effect of microservice architecture fault location, a method of microservice architecture fault root cause location based on knowledge map technology is proposed. Based on multidimensional association analysis algorithm, a multidimensional data classification model for microservice systems is constructed to determine whether a failure occurs. It builds a connection topology diagram of the microservice architecture, uses the PageRank algorithm of the knowledge map to dynamically adjust the weight relationship of each service component in the topology diagram, calculates the service component with the highest probability of failure as the root cause of the failure, and achieves the root cause location of the microservice architecture. The test results show that the accuracy rate of the proposed method's positioning results is higher than 0.75, the recall rate is above 0.71, and the F1 score is above 0.75. This effectively improves the fault location effect of the microservice architecture, with strong practicality and reliability.

Keywords: knowledge graph; microservice architecture; root-cause localization; PageRank algorithm

0 引言

微服务架构是一种新兴的企业软件架构，旨在提高企业应用程序的可扩展性和内在可维护性。微服务架构鼓励将一个应用程序分解为多个单独的服务，每个服务处理不同的相关任务，使开发人员更容易地实现服务之间的集成，更快地构建应用程序，以及对于安全性和可维护性的提升。但微服务架构也不可避免地会出现故障，使得故障的发生率比其他架构模式更高，还有可能出现因构架内部服务问题而导致系统性能下降的情况。

近年来，有很多机构和学者开展微服务架构中故障根因定位方法的研究。邹丹丹等人[1]分析微服务系统的分布式特征，提出了一种基于调用链双向搜索的故障根因定位算法。自顶向下统计故障信息，自底向上进行分层级根因定位，输出Top K根因节点，实现微服务架构故障根因定位。于庆洋等人[2]针对调用链结构复杂的问题，提出基于调用链控制流分析的微服务架构中故障根因定位方法。将大量调用链结构聚合为少量方法调用模型；并提出基于调用模型的执行时间分解模型及预测方法，将实际值与预测值的相对误差超过设定阈值的待检测数据判定为异常，分析和检测微服务性能异常及其根因。

但以上故障根因定位方法均只考虑指标，或者仅对一种或两种数据进行分析处理，考虑的数据源太单一，难以从多角度、多数据源更全面地分析微服务架构在故障发生时的多数据关联效果。并且这些方法对微服务的部署架构研究不够深入，缺少部署架构和微服务的关联数据，难以保障故障根因定位结果的准确性。为此，本文提出一种基于知识图谱的微服务架构的故障根因定位方法，旨在解决微服务架构故障定位的结果不准确，故障定

*基金项目：中国海洋石油集团项目（CNOOC-JTKJ-ITC-2021-01）
收稿日期：2023-11-30

位时间长等关键性问题。首先利用多维数据关联分析的方式对某微服务系统进行数据分析，采用人工智能分类算法构建分类模型，判断出该微服务系统是否发生故障。在故障发生时采用基于知识图谱的PageRank算法对该微服务架构的各个节点进行权重设置，通过对各服务组件的权重计算，判断出发生故障的根本节点。该方法可以有效监测运行中的系统是否发生故障，故障根因定位效果显著。

1 微服务架构的故障定位

微服务故障根因定位主要解决两个核心问题：运行系统是否发生故障及故障根由。本文首先以微服务架构的方式部署待测系统，使用本公司自研的探针来实时采集该待测系统的各种服务组件的信息，包括实时运行的指标数据、系统运行的日志数据、服务组件之间联通的网络数据、服务组件之间相互引用的调用链数据等。将这些数据分别存储在关系型数据库和图数据库中，使用多维数据关联分析算法实时检测待测系统是否发生故障。在故障发生时，根据存储在图数据库中的拓扑信息，使用PageRank算法检测出发生故障的具体节点，对该节点的多种性能指标数据进行异常检测。通过异常指标种类和异常类型判断出故障类型，最终得到故障根因定位结果。

1.1 基于多维关联分析算法分析数据

多维关联分析算法主要用于实时诊断某微服务系统是否发生故障，基本原理是通过对某微服务系统的周期性黄金指标数据、基础性能指标数据、服务组件之间的调用链数据、各服务组件之间运行日志数据等进行关联分析，判断该系统是否发生故障[3-4]。多维关联分析以上述多种指标数据作为特征数据，构建故障数据集，使用人工智能分类算法例如，支持向量机(support vector machine，SVM)，随机森林(random forest，RF)，梯度提升树算法(light gradient boosting machine，LightGBM)来判断是否发生故障，具体实现原理如图1所示。

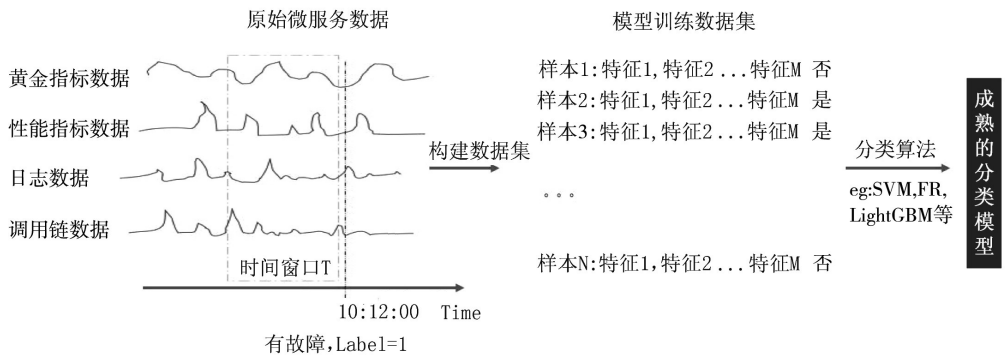


图1 多维关联分析算法的基本原理

如图1所示，多维关联分析算法是同时对微服务系统的多维度数据进行综合分析，构建模型训练的数据集。该数据集的特征来源是同一个时间窗口T中的各种原始数据，以及原始数据衍生出的新特征[5-7]。例如文本类日志数据的特征工程，可以转换为每分钟的日志条数指标及每分钟带有error关键字的日志条数指标等。将这些新的特征引入总数据集中，可以构建有效的数据特征分类模型。

在数据分类过程中，设定时间窗口内数据的近似系数为d_{i-1}，细节系数为x_{i-1}，数据的特征分析过程为：

$$f(i)=\sqrt{\frac{2}{N_k}}\sum_{i=1}^{n}x_i(k) \tag{1}$$

式中，$f(i)$表示数据i的特征提取函数；N_k表示数据分区k的数量；$x_i(k)$表示每一个数据分区的函数值。通过奇序列与偶序列的方式，对$x_i(k)$进行分类处理，两个序列的分类函数为：

$$\begin{cases}x_1(k)=\dfrac{x_{i-1}d_k}{k_i}\\ x_2(k)=\dfrac{x_i d_{k+1}}{k_i}\end{cases} \tag{2}$$

式中，$x_1(k)$表示奇序列的区域k分类函数，$x_2(k)$为偶序列区域k的分类函数；x_{i-1}和x_i分别表示故障数据的分解层数；d_k和d_{k+1}则表示两个相邻故障数据集的分类均方误差；k_i表示故障数据分类类别总数。在综合了两个序列的所有函数样本后，就可以对其进行合并处理，并综合排序。最后还需要通过不断迭代，进行数据分类与更新，计算公式为：

$$f'(i)=f(i)-f(i-1)\left(x_1(k)+x_2(k)+\frac{1}{2}\lambda_i\right) \tag{3}$$

式中，$f'(i)$表示更新后故障数据集；$f(i-1)$表示更新前的故障数据；λ_i表示近似信号的量化系数。通过以上数据，可以在一定程度上为故障数据集提供分类标签。

针对数据集的标签，可以通过本公司自研的M商城

系统和故障注入工具自动打标，其基本原理为：编写一个shell脚本，用于触发故障注入工具中的不同类型的故障。在M商城系统不同对象(比如某个主机，或某个容器，或某个服务组件)中注入故障，将故障的持续时间控制在5分钟−10分钟之间[8−10]。触发故障的时间点由脚本自动随机生成，这样就可以生成带有故障发生时间、发生地点(某个对象)、故障类型(注入故障的类型)的原始数据。对M商城系统中采集到的多种指标数据进行汇总分析，可以得到图1中的模型训练数据集，由此完成微服务系统图数据集中拓扑数据的分析，为后续故障根因定位提供数据基础。

1.2 基于知识图谱实现故障根因定位

基于上节经过分类分析的图数据集中的拓扑信息，使用PageRank算法检测出发生故障的具体节点，实现故障根因定位。PageRank算法一般用于网页重要性的排名，其基本原理是：网页的重要性取决于指向其网站的权威性。例如，如果一个新闻网站收到一个具有较高权威报纸网站的指向，那么这个新闻网站的权威性和重要性就会提高。PageRank算法在不断更新网页重要性的同时，还要考虑连接的权重[11−12]。一个网页的重要性取决于指向其网页的相对重要性。此外，PageRank算法还为每个网页分配一个数字，它一般用于计算搜索引擎结果中网页的排名。

在根因分析算法中借鉴PageRank算法的核心原理，将其应用于知识图谱中各节点重要性的计算上。计算流程为，首先根据配置管理数据库(configuration management database，CMDB)中的拓扑关系初始化每个服务组件的权重，其次根据调用该服务组件的个数和调用方的重要性来提高或降低权重。

PageRank算法会使用到系统的静态配置信息以及在系统运行过程中产生的动态数据，如各节点性能指标、调用链、日志等。CMDB是一个逻辑数据库，包含了配置项全生命周期的信息以及配置项与数据库间的物理关系、实时通信关系、非实时通信关系、依赖关系，CMDB配置信息主要作用是展现系统节点调用、组成、包含等关系。使用CMDB配置信息构建带权有向拓扑图如图2所示。

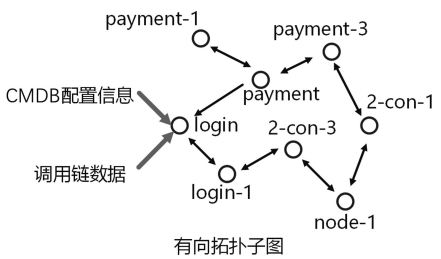


图2 带权有向拓扑图构建过程

如图2所示，可以根据实际环境或者测试结果调整有向边的权重等级和对应数值，但整体调整思路无法变化，即异常程度越高的调用关系所对应的有向边的权重越高。

举例说明，若存在节点A、B、C，对于节点A和B，如果节点A调用了节点B的服务，则创建有向边A−>B，并设置边的权重等级为2。

对于节点A、B和C，如果节点A的服务是由节点B和C组成，则创建有向边A−>B，A−>C，并设置边的权重等级为2；创建有向边B−>A，C−>A并设置权重等级为1。

对于节点A，B和C，如果节点A包含了节点B和C，比如说容器B和C部署在主机A上，或者主机上启动了进程B和进程C，则创建有向边A−>B，B−>A，A−>C和C−>A，并设置权重等级为2。

基于调用链数据计算调用关系(有向边)的异常分数，根据计算结果调整边的权重等级。当一条有向边同时具有多个调用关系时，可以将多个调用异常分数的均值作为最后的异常分数，并且有向边的调用关系数量也可以作为调整边权重等级的因素。

使用节点的性能指标及日志数据来计算节点的异常分数，构建出用于PageRank算法的输入向量。对于不同类型的节点，可以筛选合适的性能指标用来检测并计算异常分数，如主机节点异常分数的检测与计算，可以挑选CPU使用率、内存使用率、磁盘使用率等性能指标。服务节点异常分数的检测与计算，可以挑选响应时间、响应成功率、请求数等指标。对于性能指标，可以使用KDE、SHEDD、差分等简单的异常检测算法计算指标的异常分数，在计算节点的异常分数时可以取该节点下所有指标的异常分数均值、最大值或中位值。对于日志数据，可以通过定位关键词的方式来计算节点的异常分数，如日志语句中记录某个节点在某个时刻发生了error、core fault或者out of memory等异常，则该节点会有较高的异常分数。即用不同关键词对应不同异常等级来定量地计算节点的异常分数，如error对应严重异常，warn对应一般异常。

节点关系是知识图谱的中心及处理问题的关键。关系构建包括：实体与实体及实体与属性之间的关系。依据节点故障关系数据库里的保存记录，实体和实体之间的关系可直接在本体模型中承袭。

关系模型公式(4)为：

$$R_{ij\times mn}=\left(f'(i)_{ij},f'(i)_{mn}\right) \tag{4}$$

式中，实体间关系模型用$R_{ij\times mn}$描述，第i类及m类故障节点在数据库$f'(i)$所属本体中的位置用n及j描述。

在故障节点关系模型的基础上，利用参数阈值筛选潜在的故障根因节点，并将选中节点的异常分数排列表

示成向量。

构建完带权有向拓扑图和输入向量后，需应用 PageRank 算法搜索故障根因节点。首先，将输入向量 b 进行归一化处理，使向量的数值和为 1。基于输入向量的节点，从拓扑图中挑选对应的节点，构成拓扑子图，可以表示为关联矩阵 M，将关联矩阵 M 的列向量进行归一化处理，保证列向量的数值和为 1。最后设置 PageRank 算法的超参数 α 和 round，调用 PageRank 算法的公式计算最后的根因得分，计算公式如下：

$$b=\alpha\times R_{ij\times mn}\times M+(1-\alpha)e \tag{5}$$

式中，e 是均值向量，α 是随机跳转概率，M 是迭代次数。向量 b 中每一个值对应的节点即是根因节点的概率值，概率值越高则该节点为根因节点的可能性越大。可以设置阈值筛选根因节点，或者直接挑选概率值最大的 N 个节点作为最终根因故障定位结果，实现故障根因定位。

2 测试与结果分析

为了检测上述根因定位方法的有效性与定位过程的时效性，制定了如下实验。

2.1 测试准备

本文所采用的待测系统是本公司自研的 M 商城系统，其包含的服务组件如表 1 所示。

表 1 服务组件名称

服务组件名称	个数	说明
Nacos	1	用于存储整个 M 系统的默认配置
MySQL	1	用于存储 M 系统产生的各种页面数据,中间数据等
ElasticSearch	2	用于存储 M 系统中各种组件运行的日志数据
Kibana	1	用于展示 ElasticSearch 组件中存储的数据
Kafka	2	用于处理 M 系统中各种消息流转,消息代理等。
Nginx	1	用于负载均衡,消息分发
M 系统前端	1	用于 M 商城系统的前端页面展示
M 系统后端	2	用于 M 商城系统的后端逻辑处理。

本文所采用的数据采集工具是公司自研的一体化智能探针，其主要作用是实时地从各种服务组件、主机、容器、网关等设备中采集实时指标数据和服务组件之间的调用链数据。通过分布式系统跟踪技术进行数据关联并跨系统进行事务跟踪，获取事务端到端流程的性能参数和可用信息。

本文所采用的故障注入工具是公司自研的 T 系统，其包含前端页面和后端故障注入程序。可以在前端页面上选择故障类型的同时，选择待测系统的主机，在待测系统中注入特定故障，将故障注入时间控制在 5 分钟或 10 分钟，确保故障持续时间的可控性。可选的注入故障类型有：主机方面的 CPU 陡增故障、CPU 爬升故障、内存过高故障、磁盘不足故障、磁盘读 IO 故障、磁盘写 IO 故障。容器方面的 CPU 陡增故障、CPU 爬升故障、内存过高故障、进程中止故障。

根因定位系统是对待测系统进行实时故障检测，并在故障发生时进行实时根因定位的系统，其主要数据流程如图 3 所示。

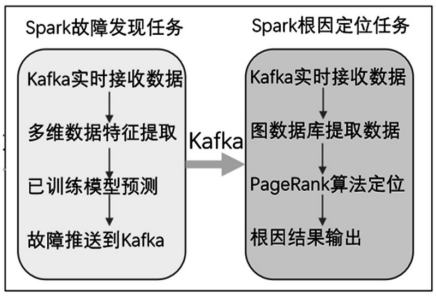


图 3 根因定位系统的整体数据流程

2.2 测试过程

(1) 使用微服务架构方式部署待测系统—M 商城系统；

(2) 在 M 商城系统所在的每个主机节点和每个容器中安装一体化智能探针，确保可以实时采集到数据，并将数据上报到数据中台和 Kafka 消息中间件；

(3) 将根因定位系统对接 Kafka 消息中间件，获取实时的流式数据，此时没有故障，不会进行根因分析；

(4) 使用故障注入工具 T 系统向 M 商城系统中注入故障，每次注入一种故障，每种故障持续 5 分钟—10 分钟不等，并将故障注入的时间和类型、注入对象等信息记录到 MySQL 表中；

(5) 使用根因定位系统实时检测是否发生故障，且给出故障发生的组件名称和故障类型；

(6) 将根因定位系统输出的根因结果与注入的故障类型进行比对，计算出根因定位算法的准确率和召回率，计算出 F1 得分；

(7) 统计耗时结果，通过修改 spark 集群中多维关联分析算法任务和知识图谱根因定位算法任务的参数(主要修改每个任务的 driver 和 executor 的 CPU 核数，内存占用大小)，计算出每种算法任务给出结果的耗时。

2.3 结果分析

使用上述基于知识图谱的故障根因定位方法，对注入的故障进行准确性测试，可以得到如表 2 所示的测试结果。

如表 2 所示，对于注入的故障，所提方法定位结果的 F1 得分均在 0.75 以上，召回率在 0.71 以上，准确率高于 0.75。表明在大部分主机和容器的故障中，所提方法具有较好的定位效果。另外，对比主机类故障的结果和容器类故障的结果，发现两者差异性不明显，表明该算法具有较好的普适性。

(下转第 89 页)