MicroAFL:一种云上微服务故障自动定位方法

羊麟威[1],李　静[1],饶涵宇[2],高　颖[3],毛　冬[2],乔宇杰[3]
(1. 南京航空航天大学 计算机科学与技术学院,江苏 南京 211106;
2. 国网浙江省电力有限公司 信息通信分公司,浙江 杭州 310016;
3. 国家电网有限公司 信息通信分公司,北京 100761)

摘　要:随着云上微服务系统规模的不断扩大,微服务之间的依赖关系变得更加紧密复杂,某个微服务的故障可能会通过微服务之间的互相调用传播至其他微服务,进而导致整个微服务系统发生异常。面对依赖关系复杂的微服务系统,考虑到故障的传播性,设计了一种云上微服务故障自动定位方法 MicroAFL。首先,MicroAFL 实时监测与收集微服务系统运行指标数据,基于自编码器模型对运行指标数据进行分析,判断微服务系统是否存在异常;一旦检测到异常,MicroAFL 通过解析云上微服务运行实例之间的通讯数据获取微服务之间的调用关系,进而构建服务调用关系图以刻画故障传播途径;其次,将各个微服务的运行状态与系统资源利用率相关联从而计算服务调用关系图中每个节点的异常权重,并通过改进的加权 PageRank 算法推断和定位引发异常的故障微服务;最后,在华为云上搭建名为 Sock-shop 的微服务系统对 MicroAFL 的故障定位准确性进行评估,实验结果表明 MicroAFL 的故障定位准确率相较对比方法有所提升。

关键词:自编码器;微服务;云环境;故障自动定位;服务调用关系图;故障传播
中图分类号:TP311　　**文献标识码**:A　　**文章编号**:1673-629X(2023)05-0088-08
doi:10.3969/j.issn.1673-629X.2023.05.014

MicroAFL:Automatic Fault Location for Microservices on Cloud

YANG Lin-wei[1],LI Jing[1],RAO Han-yu[2],GAO Ying[3],MAO Dong[2],QIAO Yu-jie[3]
(1. School of Computer Science and Technology,Nanjing University of Aeronautics and Astronautics,
Nanjing 211106,China;
2. Information and Communication Branch of State Grid Zhejiang Electric Power Company,
Hangzhou 310016,China;
3. Information and Communication Branch of State Grid Corporation,Beijing 100761,China)

Abstract:With the expansion of the scale of microservice system on the cloud,the dependencies between distributed components of microservices become more complex. The fault of a microservice may be propagated to other microservices through the mutual calls of microservices,which will lead to the entire microservice system. With the complex dependencies of microservices system and the propagation of faults,we design MicroAFL,an automatic fault location for microservices on cloud. Firstly,MicroAFL monitors and collects the metric data of the microservice system in real time,analyzes the metric data based on the autoencoder model,and judges whether there is any abnormality in the microservice system. Once an anomaly is detected,MicroAFL obtains the calling relationship between microservices by analyzing the communication data between the running instances of the microservice on the cloud,builds a microservice calling relationship diagram to describe the fault propagation path. Then,the running status of each microservice is associated with the system resource utilization to calculate the anomaly weight of each node in the microservice call graph,and the improved weighted PageRank algorithm is used to infer and locate the faulty microservice that caused the anomaly. Finally,a Sock-shop microservice system was built on Huawei Cloud to evaluate the fault location performance of MicroAFL. The experimental results show that the fault location accuracy of MicroAFL is improved.

Key words:autoencoder;microservice;cloud environment;automatic fault location;service call diagram;fault propagation

收稿日期:2022-07-03　　**修回日期**:2022-11-09
基金项目:国家电网有限公司科技项目(5700-202152169A-0-0-00)
作者简介:羊麟威(1997-),男,硕士,研究方向为数据挖掘;通讯作者:李　静(1976-),女,副教授,博士,CCF 会员(54064M),研究方向为数据挖掘、图像处理。

0 引　言

随着云计算、移动计算等不同计算方式的出现,微服务架构[1]成为软件服务设计、开发和交付的最新趋势,越来越多的互联网企业采用微服务架构开发、部署分布式应用软件。例如,腾讯的微信系统包含 3 000 多个微服务模块,运行在 20 000 多台物理主机上[2]。但微服务体系结构带来便捷开发的同时,由于其复杂的依赖关系以及频繁交付和部署,导致系统面临更多发生故障的潜在威胁[3],系统随时可能出现意想不到的故障,如并发异步错误、运行资源短缺错误等,当某一微服务模块发生故障时,相关模块组件也会因为依赖调用而发生故障,从而导致大规模的微服务级联故障。为了保证微服务的可靠运行和服务质量,开发人员必须快速修复系统故障。

然而,在微服务体系架构中定位故障会遇到以下挑战:(1)复杂的依赖关系。在微服务架构中,微服务的数量通常多达上百或者上千个,并且通常分布在多台服务主机上,服务之间的调用和依赖复杂且动态变化,一个服务的性能下降可能会广泛传播,导致多个服务出现异常。(2)大量的监控指标。大规模服务之间的通信和调用产生了大量的指标,从中分析出微服务发生异常的每个指标阈值是非常耗时的。(3)频繁的微服务更新。为满足用户需求,需要经常更新微服务模块,在更新过程中,旧模块被新服务所替代,服务之间的依赖关系也会随着更新而变化,从而形成一个动态的体系架构。

针对以上问题,该文提出一种基于自编码器的云上微服务故障定位方法:MicroAFL。MicroAFL 是一个基于容器微服务非侵入式程序模型,无需对微服务系统本身作任何修改。MicroAFL 首先通过编码器检测系统异常,一旦检测到系统异常,MicroAFL 通过微服务之间的调用关系构建服务调用关系图,并将微服务的性能状态与系统资源利用率相关联从而计算服务调用关系图中每个节点的异常权重,接着通过加权的 PageRank 算法对故障微服务进行排序,完成故障自动定位。实验表明,MicroAFL 在故障定位精度上优于对比方法。

该文的贡献如下:

(1)提出基于自编码器重构误差的微服务异常检测方法,学习微服务运行状态与响应时间波动的关联关系,实时检测微服务运行状态。

(2)通过解析微服务之间的通信数据,捕获微服务之间的调用关系,从而构造服务调用关系图以模拟故障传播路径,结合自编码器对微服务响应时间的重构误差与系统资源的利用率更新服务调用关系图中节点的异常权重。

(3)基于服务调用关系图中节点的异常权重关系,提出加权的 PageRank 算法,增强随机游走概率与节点异常程度的关联性。

(4)通过在华为云上的 CCE 集群中搭建的 Sock-shop 微服务系统中注入不同类型的故障检测 MicroAFL 性能,实验结果表明,MicroAFL 在故障定位精度上相较对比方法有所提升。

1 相关工作

微服务系统在运行过程中某个微服务可能发生网络延迟、系统资源分配不足等问题,进而引起该微服务运行异常,最终导致整个系统异常,无法对外提供正常服务。该文将检测异常微服务和定位引发系统异常的故障微服务的过程称之为故障定位。目前,学术界和工业界已提出许多解决方案来实现分布式系统中的故障定位,主要分为以下三类。

1.1 基于日志分析的方法

日志是由代码运行过程中输出的非结构化信息,记录了系统运行的完整信息,是故障诊断常见的信息来源。Xu 等人[4]通过挖掘不同组件的系统日志来构建每次请求的执行路径,根据执行路径差异诊断故障,该方法往往通常针对特定微服务系统,诊断结果往往依赖于日志质量,因此不具有很好的通用性。Nandi 等人[5]从原始日志中挖掘模板,并从挖掘的模板中提取序列特征以形成跨越分布式组件的控制流图,通过标记与运行时日志预期行为的偏差进行故障定位。然而,随着微服务系统复杂性的不断增加,日志规模不断扩大,导致这类方法在分析微服务异常方面的效率较低。

1.2 基于执行轨迹分析的方法

执行轨迹指的是利用 Pinpoint[6]等工具准确记录程序执行路径、调用关系等信息,生成类似图结构的链路追踪信息。Gan 等人[7]通过收集调用链路数据信息学习服务行为模式,主动检测系统运行过程中出现与正常服务相违背的行为来分析故障原因。Mi 等人[8]通过跟踪执行路径来收集信息,通过分析路径上的延迟偏差确定故障根因。陈皓等人[9]通过收集并建模微服务的追踪信息,将未知故障的执行追踪信息与已知故障的执行追踪信息相匹配,采用字符串编辑距离衡量相似度以诊断可能的故障原因。但是这类方法往往需要将监测代码注入到目标系统以获得准确调用链信息,面对一个由多个开发团队使用不同语言开发的微服务系统而言,在系统中加入监测点的开销成果过高,可用性较低。

1.3 基于性能指标分析的方法

通过调用操作系统或应用系统提供的接口可以采

集性能指标数据来分析系统性能变化情况,从而诊断故障根因[10]。Lin 等人[11]实时收集性能指标数据并与历史数据相结合,将表现异常的微服务模块添加到故障微服务候选集中,并根据异常微服务和性能指标之间的相关性对异常微服务排序。Thalheim 等人[12]构建指标缩减框架和指标依赖关系提取器,通过结合指标维度来推断组件之间的依赖关系进行根因定位。Gulenko 等人[13]通过收集不同微服务的性能指标数据构建对应向量,利用聚类算法训练对应微服务性能指标正常特征空间。在实时监测过程中,检查对应性能指标数据构成的向量是否处于正常特征空间来检测微服务是否发生性能故障。Liu 等人[14]通过跟踪固定监测数据分析异常微服务,接着通过拓扑图的传播方向分析异常传播链,异常传播链末端的微服务被视为可能的故障根因。时间窗口中的服务依赖关系构建拓扑图,结合历史监测数据分析异常微服务,接着通过拓扑图的传播方向分析异常传播链,异常传播链末端的微服务被视为可能的故障根因。

2 云上微服务故障自动定位方法

该文提出一种云上微服务故障自动定位方法,其总体框架如图 1 所示,主要包括 3 个模块。数据收集模块负责收集应用程序指标和系统级别指标数据,其中应用程序指标用于检测应用性能问题,系统级别指标用于后续服务调用关系图中节点权重更新。异常检测模块通过自编码器对应用程序指标数据进行编码重构,检测微服务模块是否发生异常。一旦检测到系统异常,故障定位模块通过解析微服务之间的调用关系构造服务调用关系图以分析异常传播路径,利用系统资源利用率与微服务性能的相关性计算服务调用关系图中每个微服务节点的异常权重,最终利用改进的加权 PageRank 算法定位故障根因微服务模块。

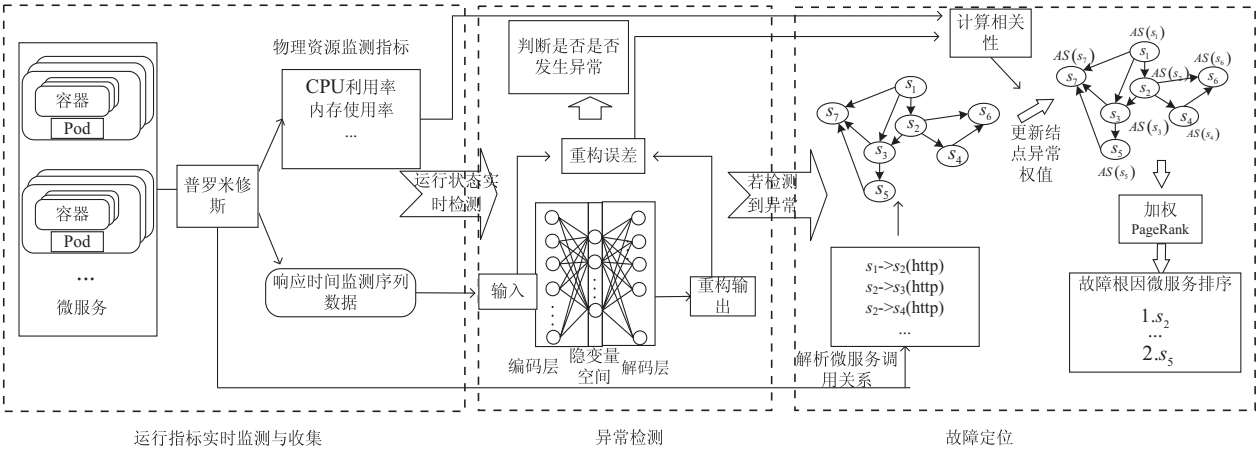

图 1 MicroAFL 总体框架

2.1 运行指标实时监测与收集

微服务系统运行监测是异常检测及故障诊断的基础,MicroAFL 作为一个非侵入式故障定位方法,无需对微服务系统进行代码注入便可实时监测并收集微服务系统运行指标数据。该文实时监测的指标主要分为物理资源利用率指标和微服务之间调用的响应时间指标。

物理资源利用率指标是反映运行微服务实例的物理机或虚拟机层面资源使用情况的一类指标,例如 CPU 利用率、内存使用率等。该文将 k 项物理资源监测指标表示为 $\boldsymbol{M}=\{m^1,m^2,\cdots,m^k\}$,将任意监测指标 m^j 的持续监测值表示为时间序列数据 $\boldsymbol{X}^j=[x_1^j,x_2^j,\cdots]$,其中 $m^j\in\boldsymbol{M},j\in\{1,2,\cdots,k\}$。为了建模指标 m^j 监测值的变化关系,在监测指标收集时刻 t,通过一个时间长度为 h 的滑动窗口 w 截取指标 m^j 的监测序列值,表示为 $\boldsymbol{X}_t^j=[x_{t-h+1}^j,\cdots,x_{t-1}^j,x_t^j]$,其中 x_t^j 表示指标 m^j 在 t 时刻的监测值。将 k 项物理资源监测指标在 t 时刻收集的包含一个时间窗口 w 的监测序列数据构成监测指标数据矩阵 $\boldsymbol{D}_w=[\boldsymbol{X}_t^1,\boldsymbol{X}_t^2,\cdots,\boldsymbol{X}_t^k]$。

响应时间指标是反映微服务系统中微服务对其他微服务请求做出的应答所花费时间长短的指标。在拥有 n 个微服务模块的微服务系 $V=\{v_1,v_2,\cdots,v_n\}$ 中,对于任意微服务模块 $v_e\in V$,在 t 时刻收集的包含一个时间窗口 w 的响应时间监测序列数据表示为 $\boldsymbol{P}_t^e=[p_{t-h+1}^e,\cdots,p_{t-1}^e,p_t^e]$,其中 p_t^e 表示微服务 v_e 在 t 时刻响应时间的监测值。将 n 个微服务模块在 t 时刻收集的历时一个时间窗口 w 的响应时间监测序列数据构成矩阵 $\boldsymbol{I}_w=[\boldsymbol{P}_t^1,\boldsymbol{P}_t^2,\cdots,\boldsymbol{P}_t^n]$。

2.2 异常检测

对微服务系统的异常检测是基于分析每个微服务的响应时间监测序列数据重构编码误差来实现的。例如对于微服务 $v_e\in V$,将在 t 时刻收集的历时一个时间窗口 w 的响应时间监测指标序列数据 $\boldsymbol{P}_t^e\in R^h$ 作为训练完成的自编码器的输入,通过编码层将 $\boldsymbol{P}_t^e$ 映射为

潜在特征表示 $\overline{\boldsymbol{P}_t^e} \in R^h$:

$$\overline{\boldsymbol{P}_t^e} = g(\boldsymbol{W}_1^{\mathrm{T}} \boldsymbol{P}_t^e + \boldsymbol{b}) \tag{1}$$

其中, g 是激活函数, $\boldsymbol{W}_1^{\mathrm{T}} \in R^{h\times d}$ 是输入层与隐藏层的权重矩阵, $\boldsymbol{b} \in R^{h\times 1}$ 是输入层的偏置向量。接着通过解码器将潜在特征表示 $\boldsymbol{P}_t^e$ 重构为微服务模块 v_e 的响应时间指标监测序列数据 $\boldsymbol{P}_t^{e'}$:

$$\boldsymbol{P}_t^{e'} = g(\boldsymbol{W}_2^{\mathrm{T}} \overline{\boldsymbol{P}_t^e} + \boldsymbol{c}) \tag{2}$$

其中, $\boldsymbol{W}_2^{\mathrm{T}} \in R^{d\times h}$ 是隐藏层和输出层的权重矩阵, $\boldsymbol{c} \in R^{d+1}$ 是隐藏层的偏置向量, $\boldsymbol{P}_t^{e'} = [p_{t-h+1}^{e'}, \cdots, p_{t-1}^{e'}, p_t^{e'}] \in R^h$ 。接着计算 $\boldsymbol{P}_t^e$ 与 $\boldsymbol{P}_t^{e'}$ 之间的重构均方误差 $J(\boldsymbol{P}_t^e, \boldsymbol{P}_t^{e'})$:

$$J(\boldsymbol{P}_t^e, \boldsymbol{P}_t^{e'}) = \frac{1}{h} \sum_{i=t-h+1}^{t} |p_i^{e'} - p_i^e|^2 \tag{3}$$

在自编码器训练阶段,使用微服务系统正常运行时微服务 v_e 的响应时间监测序列作为训练数据训练自编码器模型。经过多轮训练之后,收敛的自编码器模型学习到了正常响应时间序列数据的特征。因此,自编码器模型对微服务 v_e 正常运行时的响应时间监测值的重构值会接近于监测值,对应重构误差较小并处于一个稳定范围内波动。通过计算此时重构误差的均值 μ 以及标准差 σ ,确定微服务模块 v_e 的异常检测阈值 $\alpha^e = \mu + 3\sigma$ 。在微服务 v_e 运行状态实时检测过程中,若发现重构误差 $J(\boldsymbol{P}_t^e, \boldsymbol{P}_t^{e'}) > \alpha^e$,则认为微服务 v_e 发生异常。

2.3　故障定位

一旦检测到系统异常,MicroAFL 开始定位引发异常的故障微服务。针对微服务系统复杂的依赖调用关系,通过构建服务调用关系图可以有效地展示服务之间的依赖关系,从而刻画故障在微服务之间的传播途径。构建服务调用关系图的步骤如下:首先,MicroAFL 将微服务系统中的微服务集合记为 $V = \{v_1, v_2, \cdots, v_n\}$,其中 n 表示微服务个数。对于任意 $v_i \in V$,映射生成图节点 s_i ,最终得到图节点集合 $S = \{s_1, s_2, \cdots, s_n\}$;然后,通过解析各个微服务之间的通信数据来捕捉微服务之间的调用关系,若微服务 v_i 向微服务 v_j 发送服务请求,则构造一条从 s_i 指向 s_j 的有向边 e_{ij} ,构成边集合 $E = \{e_{ij}\}$,$1 \leqslant i,j \leqslant n$,相同服务请求只会构造一条有向边;接着,将微服务模块 v_e 的响应时间监测指标的重构误差作为节点 s_e 的初始异常权重 $f(s_e)$ 。遍历计算每个微服务模块的异常初始权重,得到节点异常权重集合 $F = \{f(s_e)\}$,$s_e \in S$;最终,得到服务调用关系图 $G(S,E,F)$ 。

以服务调用关系图 $G(S,E,F)$ 为基础,根据服务调用关系图中相邻节点间异常权重关系[15]自动更新每个节点的异常权重[16]。对于任意节点 $s_e \in S$,将包含指向节点 s_e 的有向边的相邻节点构成集合 $\mathrm{AN}(s_e)$,将包含指向 $5\mathrm{AN}(s_e)$ 中任一节点的有向边的相邻的节点构成集合 $\mathrm{NAN}(s_e)$ 。例如图 2 中的节点 s_3, $\mathrm{AN}(s_3) = \{s_1, s_5, s_6\}$, $\mathrm{NAN}(s_3) = \{s_2, s_4, s_8, s_9\}$ 。

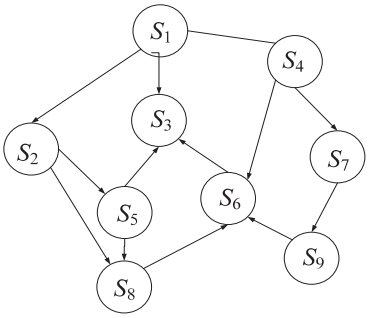

图 2　AAN 和 NHAN 集合示例

接着,计算 $\mathrm{AN}(s_e)$ 的平均异常权重 $\mathrm{aScore}(s_e)$:

$$\mathrm{aScore}(s_e) = \frac{\sum_{s_i \in \mathrm{AN}(s_e)} f(s_i)}{\mathrm{inDegree}(s_e)} \tag{4}$$

其中, $f(s_i)$ 表示节点 s_i 的异常权重, $\mathrm{inDegree}(s_e)$ 表示节点 s_e 的入度。计算 $\mathrm{AN}(s_e)$ 的平均异常权重 $\mathrm{cScore}(s_e)$:

$$\mathrm{cScore}(s_e) = \frac{\sum_{s_i \in \mathrm{NAN}(s_e)} f(s_i)}{\sum \mathrm{inDegree}(s_i)} \tag{5}$$

其中, $\mathrm{aScore}(s_e)$ 反映了 $\mathrm{AN}(s_e)$ 整体上的异常程度。 $\mathrm{cScore}(s_e)$ 表示 $\mathrm{NAN}(s_e)$ 整体上的异常程度。结合 $\mathrm{aScore}(s_e)$ 和 $\mathrm{cScore}(s_e)$ 的特征计算节点 s_e 的异常权重 $\mathrm{acScore}(s_e)$:

$$\mathrm{acScore}(s_e) = \mathrm{aScore}(s_e) - \mathrm{cScore}(s_e) \tag{6}$$

显然, $\mathrm{acScore}(s_e)$ 综合了节点 s_e 自身及其周围节点的异常信息, $\mathrm{acScore}(s_e)$ 越高,表明节点 s_e 相邻节点整体异常程度高,而 $\mathrm{AN}(s_e)$ 的相邻节点整体异常程度低,那么节点 s_e 对应的微服务 v_e 是故障根因的可能性越高。此外,微服务的响应时间与该微服务部署所在主机性能指标变化相关,该文选择皮尔逊相关函数来衡量微服务 v_e 上的响应时间重构误差 $J(\boldsymbol{P}_t^e, \boldsymbol{P}_t^{e'})$ 和部署微服务 v_e 的主机上各项监测指标 $\boldsymbol{X}^j, j \in \{1,2,\cdots,k\}$ 的相关性,表示为 $\mathrm{Corr}(J(\boldsymbol{P}_t^e, \boldsymbol{P}_t^{e'}), \boldsymbol{X}^j)$ 。微服务 v_e 的运行状态可以通过 $\mathrm{Corr}(J(\boldsymbol{P}_t^e, \boldsymbol{P}_t^{e'}), \boldsymbol{X}^j)$ 得分高低来体现[16],因此,结合 $\mathrm{acScore}(s_e)$ 与 $\mathrm{Corr}(J(\boldsymbol{P}_t^e, \boldsymbol{P}_t^{e'}), \boldsymbol{X}^j)$ 计算节点 s_e 的最终异常权重 $\mathrm{AS}(s_e)$ 为:

$$\mathrm{AS}(s_e) = \mathrm{acScore}(s_e) \times \frac{1}{k} \sum_{j=1}^{k} \mathrm{Corr}(J(\boldsymbol{P}_t^e, \boldsymbol{P}_t^{e'}), \boldsymbol{X}^j) \tag{7}$$

遍历计算每个微服务节点的异常权重完成服务调用关系图中所有节点异常权重更新。接着采用

PageRank 算法[17]在服务调用关系图 $G(S,E,F)$ 中“随机游走”,进一步对每个微服务模块作为故障根因微服务的概率大小进行排序[18]。针对微服务故障定位场景,该文提出一种改进的加权 PageRank 算法,利用节点异常权重与相连节点异常权重的关系计算游走概率,提升定位故障根因定位准确性。加权 PageRank 算法随机游走策略是基于每个节点访问其他节点的概率,因此首先需要定义服务调用关系图节点转移概率矩阵 $\boldsymbol{U}$:

$$\boldsymbol{U}=\begin{pmatrix} u_{11} & \cdots & u_{1j} \\ \vdots & \ddots & \vdots \\ u_{i1} & \cdots & u_{ij} \end{pmatrix} \tag{8}$$

其中,u_{ij} 代表从节点 s_j 随机游走至节点 s_i 的概率。节点异常权值的大小与游走概率相关[19],计算从节点 s_j 随机游走至节点 s_i 的概率 u_{ij}:

$$u_{ij}=\begin{cases} \dfrac{\mathrm{AS}(s_i)}{\mathrm{linkOut}(s_j)}, & \exists s_j \to s_i \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

其中,$s_j \to s_i$ 表示存在从节点 s_j 指向节点 s_i 的一条有向边,$\mathrm{linkOut}(s_j)$ 表示节点 s_j 指向的节点的异常权重和。对于任意节点 $s_e \in S, e=\{1,2,\cdots,n\}$,初始化 PR 分数为 $\mathrm{PR}_0(s_e)=1/n$,将所有节点的 PR 分数表示为向量 $\boldsymbol{R}_0$:

$$\boldsymbol{R}_0=(\mathrm{PR}_0(s_1),\mathrm{PR}_0(s_2),\cdots,\mathrm{PR}_0(s_n))^{\mathrm{T}} \tag{10}$$

在每轮随机游走过程中,迭代更新每个节点的 PR 分数:

$$\boldsymbol{R}_m=d\boldsymbol{U}\cdot\boldsymbol{R}_{n-1}+(1-d)\boldsymbol{R}_0 \tag{11}$$

其中,$\boldsymbol{R}_m$ 表示第 m 轮迭代后所有节点的 PR 分数向量,$\boldsymbol{U}\in \mathrm{R}^{n\times n}$ 表示随机游走概率矩阵,$d\in(0,1)$ 表示索尼系数,通常 $d=0.85$[19]。经不断迭代更新,每个节点的 PR 分数会趋于收敛,此时节点 PR 分数越高,所对应的微服务模块是故障根因的可能性越大,最终根据节点 PR 分数从高到低的顺序输出故障根因微服务排名列表。

3 实 验

3.1 运行指标实时监测与收集

3.1.1 实验环境搭建

搭建的实验环境框架如图 3 所示。将 Sock-shop 通过 Docker 部署在华为云 CCE 集群中,每个 Pod 副本最大容量设置为 3,Istio 作为服务代理与每个容器共同部署在相同 Pod 中,通过编写 locust 脚本模拟用户对 Sock-shop 的接口发起请求,利用第三方工具对 Sock-shop 注入不同类型的故障来模拟微服务系统出现不同的故障,利用 prometheus(普罗米修斯)收集每个微服务相应监测指标。

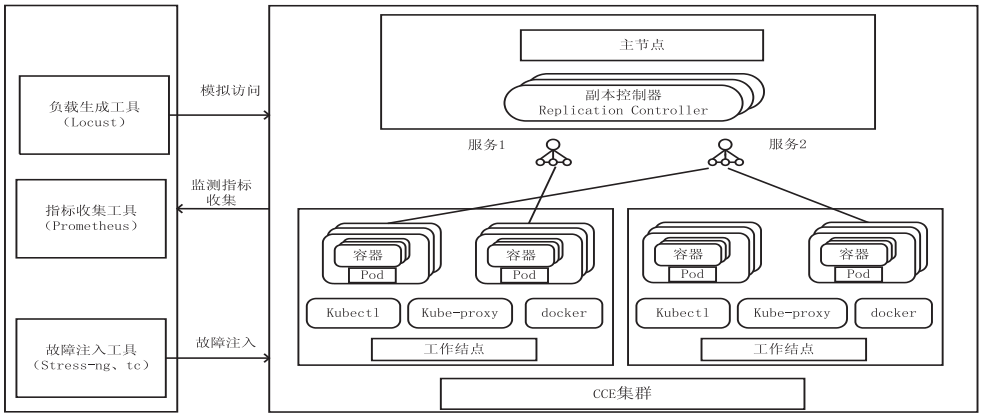

图 3 实验环境框架

搭建实验环境的主要软硬件配置信息如表 1 所示。

表 1 实验环境搭建使用的硬件和软件配置信息

名称	配置信息
华为云 CCE 集群	V1.19
弹性云服务器	4vCPUs|16GiB|c6.xlarge.4
Istio	V1.8.4-r2
Prometheus	V2.21.11

3.1.2 实验对象

Sock-Shop 是一款基于 Springboot 开发的模拟销售袜子的电子商务网站微服务应用程序,是广泛使用的微服务基准应用,微服务之间的通信以 HTTP 通信为主,并使用 RESTful 的接口设计风格。每个微服务都可独立开发、部署和扩展。

3.1.3 模拟访问

该文使用分布式开源工具 Locust 根据不同微服务接口请求规则编写对应 Locust 脚本,并发生成多个线程执行对不同微服务的请求,模拟用户访问 Sock-shop 系统从而使得各个微服务模块之间正常通信与调用请求,考虑到不同微服务模块被访问的频率存在差异,如表 2 所示,对不同微服务模块配置了不同的请求频率。

表 2　模块请求频率配置

微服务模块	并发线程数	请求频率/s
Front-end	100	20
Catalogue	100	20
User	100	10
Carts	100	10
Orders	100	10
Shipping	100	10

3.1.4　故障注入

为了模拟微服务系统在运行过程中遇到的性能故障问题,在 Sock-shop 中注入了以下三种类型的故障:(1)网络延迟(Latency):使用 tc 工具造成微服务的网络 300 ms 延迟,模拟在微服务运行过程中的网络延迟。(2)CPU 资源耗尽(CPU Hog):stress-ng 是一款占用计算机 CPU 资源的工具,通过 stress-ng 占用 CPU 资源 80%,耗尽系统 CPU 资源,模拟微服务缺少运行所需 CPU 资源故障。(3)内存泄漏(Memory Leak):使用 stress-ng 对某个微服务节点持续分配内存,模拟微服务系统因所在服务器内存不足而引发的内存泄漏故障。

3.2　对比方法和评价指标

为了评估 MicroAFL 的故障自动定位性能,选取以下方法作为实验性能对比方法:

Random selection(RS):随机选择是运维人员在不了解系统特定领域知识的情况下使用的一种方法,故障修复人员每次都会从未排查的微服务中随机选择一个微服务进行故障诊断,直到定位故障。

MicroRCA:MicroRCA[18]利用聚类算法发现异常微服务,再基于服务依赖图提取异常子图,最终使用随机游走算法进行故障定位。

AAMR:AAMR[16]首先基于实时指标数据构建服务依赖图,并通过计算每个微服务的异常分数自动更新每个微服务的异常权重。最后,应用基于页面的随机游走对根本原因进行进一步排序,即对潜在的故障根因微服务模块按照可能性大小进行排序。

为了量化算法模型的性能,采用评估指标 AC@K 和 MAP 衡量故障定位效果[18]。

AC@K 表示对根因预测输出的前 K 个微服务中包含故障根因微服务的概率。AC@K 分数越高,代表算法模型定位故障越准确。在实验中,选择 $K=1$ 和 $K=3$ 评价算法识别故障定位性能,$|A|$ 表示测试的故障用例数,$Z^a[i]$ 是每个微服务模块在故障用例 a 中根因预测中的输出排名,V_{rc}^a 是故障根因微服务的集合。AC@K 在一组异常测试集 A 上的定义为:

$$\mathrm{AC@K}=\frac{1}{|A|}\sum_{a\in A}\frac{\sum_{i<K}Z^a[i]\in V_{rc}^a}{\min(K,|V_{rc}^a|)} \tag{12}$$

MAP 量化算法的故障定位平均准确率,其中 n 是微服务的数量。MAP 在异常测试集 A 上定义为:

$$\mathrm{MAP}=\frac{1}{|A|}\sum_{a\in A}\sum_{1\leqslant K\leqslant n}\mathrm{AC@K} \tag{13}$$

3.3　故障定位实验结果

表 3 展示了注入不同类型故障下,MicroAFL 在 Sock-shop 中不同微服务上的故障定位准确率。从表 3 中可以看出,MicroAFL 在每个微服务上的故障定位平均准确率达到了 0.85 以上。此外,发现 MicroAFL 在 Shipping 微服务上的故障定位准确率普遍低于其他模块,分析这是由于相比其他模块,Shipping 微服务被请求的频率以及访问其他模快的频率较低,使得 Shipping 微服务的响应时间指标监测序列数据波动变化更小,即使由于 Shipping 微服务引发系统异常,也不易被判定为故障根因。

表 3　MicroAFL 在不同模块上的故障定位准确率

Metric		Orders	Catalogue	User	Carts	Shipping
Latency	AC@1	0.925	0.911	0.884	0.904	0.872
	AC@3	0.943	0.932	0.904	0.933	0.904
	MAP	0.932	0.924	0.894	0.916	0.885
CPU Hog	AC@1	0.878	0.865	0.839	0.851	0.842
	AC@3	0.911	0.904	0.878	0.896	0.873
	MAP	0.895	0.877	0.854	0.875	0.857
Memory Leak	AC@1	0.914	0.908	0.892	0.889	0.876
	AC@3	0.957	0.944	0.927	0.921	0.913
	MAP	0.933	0.922	0.907	0.911	0.894

3.4　比较与分析

为了比较 MicroAFL 与对比方法在故障定位准确率上的差异,复现了对比方法在本实验收集的故障注入测试用例集上的实验结果。实验结果如图 4 所示,

在注入三种故障以及 K 取不同值的情况下,MicroAFL 的故障定位准确率均高于其他方法,表明 MicroAFL 在故障定位方面准确率方面确实得到了有效提升。通过分析,故障定位的准确率提升主要归因于构建服务调用关系图时结合了异常检测时的重构误差,重构误差的大小在一定程度上反映了故障的严重程度和持续时间,在服务调用关系图节点权重更新时结合了重构误差所携带的特征。

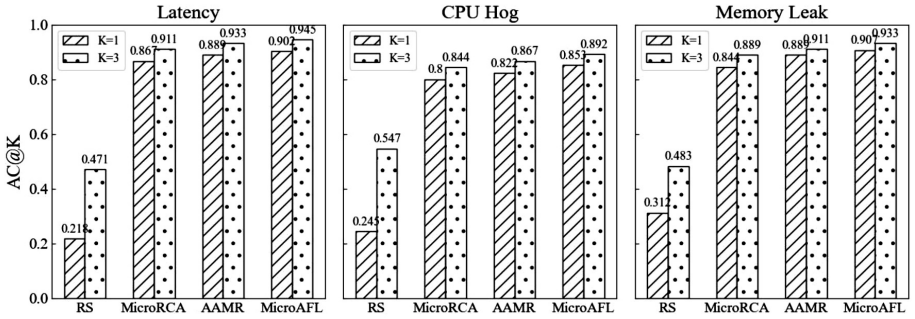

图 4 故障定位准确率对比实验结果

进一步,通过计算故障定位评价指标 MAP 衡量 MicroAFL 算法故障定位平均准确率,实验结果如图 5 所示。从图 5 中可以看出,MicroAFL 算法对网络延迟(Latency)的故障定位平均精度为 0.920,对 CPU 资源短缺(CPU Hog)故障定位的平均精度为 0.864,对内存泄漏故障(Memory Leak)定位的平均精度为 0.912,优于对比方法,证明了文中方法对于提高故障定位准确率的有效性。

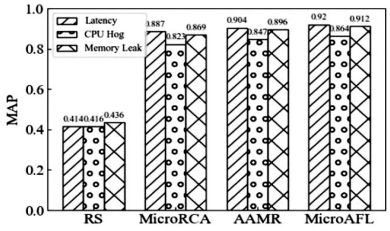

图 5 故障定位平均准确率对比结果

4 结束语

该文设计了一种名为 MicroAFL 的云上微服务故障自动定位方法,用于云环境下微服务系统异常检测和故障定位。MicroAFL 通过微服务之间的调用关系构建服务调用关系图以刻画云上具有复杂依赖关系的微服务之间的故障传播路径,进而使用一种基于异常权重加权的 PageRank 随机游走算法对故障微服务进行排序,完成故障自动定位。但该方法在节点异常权重计算过程中只利用了监测指标信息,特征提取相对单一,在以后的工作中,将在服务调用关系图构建以及节点权重计算过程中,结合日志、调用链等信息进一步分析微服务运行特征,从而研究更细粒度的故障定位。

参考文献:

[1] HASSAN S, BAHSOON R. Microservices and their design trade-offs: a self-adaptive roadmap[C]//2016 IEEE international conference on services computing (SCC). San Francisco: IEEE, 2016: 813-818.

[2] ZHOU X, PENG X, XIE T, et al. Fault analysis and debugging of microservice systems: industrial survey, benchmark system, and empirical study[J]. IEEE Transactions on Software Engineering, 2018, 47(2): 243-260.

[3] ZHANG H, LI S, JIA Z, et al. Microservice architecture in reality: an industrial inquiry[C]//2019 IEEE international conference on software architecture (ICSA). Hamburg: IEEE, 2019: 51-60.

[4] XU W, HUANG L, FOX A, et al. Detecting large-scale system problems by mining console logs[C]//Proceedings of the ACM SIGOPS 22nd symposium on operating systems principles. Haifa: ACM, 2009: 117-132.

[5] NANDI A, MANDAL A, ATREJA S, et al. Anomaly detection using program control flow graph mining from execution logs[C]//Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. San Francisco: ACM, 2016: 215-224.

[6] CHEN M Y, KICIMAN E, FRATKIN E, et al. Pinpoint: problem determination in large, dynamic internet services [C]//Proceedings international conference on dependable systems and networks. Washington: IEEE, 2002: 595-604.

[7] GAN Y, ZHANG Y, HU K, et al. Seer: leveraging big data to navigate the complexity of performance debugging in cloud microservices[C]//Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems. New York: [s. n.], 2019: 19-33.

[8] MI H, WANG H, ZHOU Y, et al. Toward fine-grained, unsupervised, scalable performance diagnosis for production cloud computing systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2013, 24(6): 1245-1255.

[9] 陈 皓,许源佳,王 焘,等. 基于相似度匹配的微服务故障诊断方法[J]. 计算机系统应用,2021,30(5):1-11.

[10] BRANDÓN Á, SOLÉ M, HUÉLAMO A, et al. Graph-based

root cause analysis for service-oriented and microservice architectures[J]. Journal of Systems and Software,2020,159:110432.

[11] LIN J J,CHEN P,ZHENG Z. Microscope:pinpoint performance issues with causal graphs in micro-service environments[C]//International conference on service-oriented computing. Hangzhou:Springer,2018:3-20.

[12] THALHEIM J,RODRIGUES A,AKKUS I E,et al. Sieve:actionable insights from monitored metrics in distributed systems[C]//Proceedings of the 18th ACM/IFIP/USENIX middleware conference. Las Vegas:ACM,2017:14-27.

[13] GULENKO A,SCHMIDT F,ACKER A,et al. Detecting anomalous behavior of black-box services modeled with distance-based online clustering[C]//2018 IEEE 11th international conference on cloud computing (CLOUD). San Francisco:IEEE,2018:912-915.

[14] LIU D,HE C,PENG X,et al. MicroHECL:high-efficient root cause localization in large-scale microservice systems[C]//2021 IEEE/ACM 43rd international conference on software engineering:software engineering in practice(ICSE-SEIP). Madrid:IEEE,2021:338-347.

[15] GE Y,JIANG G,DING M,et al. Ranking metric anomaly in invariant networks[J]. ACM Transactions on Knowledge Discovery from Data,2014,8(2):1-30.

[16] ZHANG Z,LI B,WANG J,et al. AAMR:automated anomalous microservice ranking in cloud-native environment[C]//Proceedings of the international conference on software engineering and knowledge engineering. Pittsburgh:[s.n.],2021:86-91.

[17] 黄　艳,李朝荣. 改进 PageRank 算法的网页权重分析[J]. 宜宾学院学报,2022,22(6):6-8.

[18] WU L,TORDSSON J,ELMROTH E,et al. Microrca:root cause localization of performance issues in microservices[C]//NOMS 2020-2020 IEEE/IFIP network operations and management symposium. Budapest:IEEE,2020:1-9.

[19] JEH G,WIDOM J. Scaling personalized web search[C]//Proceedings of the 12th international conference on world wide web. New York:ACM,2003:271-279.

(上接第 68 页)

[17] POONA M,ARORA S M. A DWT-SVD based robust digital watermarking for digital images[J]. Procedia Computer Science,2018,132:1441-1448.

[18] JIA Shaoli. A novel blind color images watermarking based on SVD[J]. Optik,2014,125(12):2868-2874.

[19] MOOSAZADEH M,EKBATANIFARD G. A new DCT-based robust image watermarking method using teaching-learning-based optimization[J]. Journal of Information Security and Applications,2019,47:28-38.

[20] 张绣亚,孙刘杰,王文举,等. 三维点云模型高鲁棒性多重盲水印算法研究[J]. 包装工程,2016,37(19):181-186.

[21] PAPADAKIS P,PRATIKAKIS I,PERANTONIS S,et al. Efficient 3D shape matching and retrieval using a concrete radialized spherical projection representation[J]. Pattern Recognition,2007,40(9):2437-2452.