(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 115756929 A
(43)申请公布日 2023.03.07

(21)申请号 202211470197.6

(22)申请日 2022.11.23

(71)申请人 北京大学
地址 100871 北京市海淀区颐和园路5号

(72)发明人 张齐勋 刘洪毅 杨勇 贾统 李影

(74)专利代理机构 北京万象新悦知识产权代理有限公司 11360
专利代理师 贾晓玲

(51)Int.Cl.
G06F 11/07(2006.01)

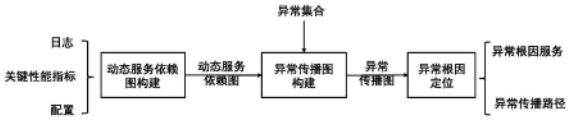
权利要求书2页 说明书8页 附图1页

(54)发明名称

一种基于动态服务依赖图的异常根因定位方法及系统

(57)摘要

本发明提供了一种基于动态服务依赖图的异常根因定位方法及系统，属于智能运维领域。本发明的异常根因定位，基于服务配置信息提取部署时服务依赖关系，基于服务间关键性能指标的相关性及日志频率信息，发现运行时服务依赖关系，动态地构建服务依赖图；基于输入的异常服务及动态服务依赖图，自动构建异常传播图；基于深度优先搜索构建异常传播路径，并计算异常的根因分数，定位异常根因服务，报告异常传播路径。本发明可以更好地捕捉运行时服务依赖关系的变化，有助于刻画更加精确的异常传播关系，提升了异常根因定位的能力；适用于各种类型异常的根因定位，具有很好的通用性；直接定位到异常的服务，并提供异常的传播路径，具有很好的实用性和可解释性。且无需人工参与，节省了人力成本。



CN 115756929 A

1.一种基于动态服务依赖图的异常根因定位方法,其特征在于,包括动态服务依赖图构建、异常传播图构建、异常根因定位;具体步骤包括:

1)动态服务依赖图构建,具体执行如下步骤:

11)提取部署时服务依赖;

12)提取运行时服务依赖;

13)构建动态服务依赖图G:每次执行异常根因定位任务时,均执行步骤11)和步骤12),动态地获取部署时服务依赖和运行时服务依赖,构建服务依赖图G=<V,E>;其中G是有向图,V表示服务,是G的节点;E表示依赖关系,是G的边;

2)异常传播图构建,具体执行如下步骤:

构建异常传播图:将输入的异常集合A=$\{a_1,a_2,...,a_n\}$标记在服务依赖图G上,将标为异常的服务节点取出,形成G的子图G′,同时保留节点间的边,节点的值为异常发生时刻的倒数;

3)异常根因定位,具体执行如下步骤:

31)构建异常传播路径:在异常传播图G′上,随机选择一个异常的服务s'_0出发,利用深度优先搜索,寻找候选根因节点及其异常传播路径;

32)计算异常根因分数:对于s'_0,利用异常传播图G′及异常传播路径Path(s'_0,s'_t),计算每一个s'_t∈R的异常根因分数;

33)报告异常根因及传播路径:对异常根因按照分数进行降序排列,并报告与之对应的异常传播路径。

2.如权利要求1所述的基于动态服务依赖图的异常根因定位方法,其特征在于,步骤11)中定义待收集配置信息的属性集合K=($k_1,k_2,...,k_n$),使得收集到的配置信息c_i=(k_i,b_i)必有k_i∈K,其中,k_i表示配置信息的属性,v_i表示配置信息的值,记一个服务s_i的配置集合为C_i=($c_1,c_2,...,c_n$),若服务s_i与服务s_j的配置集合存在交集,即$C_i \cap C_j \neq \Phi$,则服务s_i与服务s_j之间存在部署依赖。

3.如权利要求1所述的基于动态服务依赖图的异常根因定位方法,其特征在于,步骤12)采用PC算法,通过条件独立性检验判断变量之间的因果关系,再利用d-分离条件确定因果关系之间的方向,使用的条件独立性检验方法是G^2条件交叉熵度量,如式(1)所示,其服从自由度为D的x^2分布,如式(2)所示:

$$G^2 = 2mCE(X,Y|Z) = \sum_z P(z) \sum_x \sum_y P(x,y|z) \log \left(\frac{P(x,y|z)}{p(x|z)\cdot p(y|z)}\right) \qquad 式(1)$$

$$D = (N_X-1)(N_Y-1)\prod_{Z'\in Z} N_{Z'} \qquad 式(2)$$

其中,$X \epsilon R^{Nx}$,$Y \epsilon R^{Ny}$,$Z' \epsilon R^{Nz'}$,Z是Z′的集合,m是采样的个数。

4.如权利要求1所述的基于动态服务依赖图的异常根因定位方法,其特征在于,步骤13)中构造规则为,若服务s_i与服务s_j之间存在运行时依赖,则添加s_i与s_j之间的有向边,边的权重为s_i与s_j之间的G^2值,若服务s_i与服务s_j之间存在部署时依赖,则添加s_i到s_j及s_j到s_i的有向边,边的权重均为运行时依赖边权的均值。

5.如权利要求1所述的基于动态服务依赖图的异常根因定位方法,其特征在于,步骤2)中a_i=(s_i,t_i)表示第i个异常,s_i表示发生异常a_i的服务,t_i表示发生异常a_i的时刻,将异常发生的时刻进行标准化,设最先发生异常的时刻为1;其余异常按照发生的顺序,发生时刻

在1的基础上递增。

6.如权利要求1所述的基于动态服务依赖图的异常根因定位方法，其特征在于，步骤32)中计算公式如式(3)所示：

$$score(s_t') = \sum_{k=1}^{M} \prod_{i=1}^{N} w_{k,i} * s_i \qquad \text{式（3）}$$

其中，score(s′$_t$)表示异常服务s′$_t$的异常根因分数，M表示s′$_0$到s′$_t$的异常传播路径个数，N表示传播路径o_k中的跳数，$w_{k,i}$表示第k条传播路径中节点i与节点i-1之间的依赖权重，s_i表示节点i的值，即异常发生时刻的倒数。

7.一种基于动态服务依赖图的异常根因定位系统，其特征在于，包括动态服务依赖图构建模块、异常传播图构建模块、异常根因定位模块，其中：

动态服务依赖图构建模块，包括部署时服务依赖关系发现器、运行时服务依赖关系发现器、动态服务依赖图构建器、服务依赖图存储器；部署时服务依赖关系发现器使用服务配置信息，发现微服务之间的部署时依赖关系；运行时服务依赖关系发现器使用服务运行日志及关键性能指标，发现微服务之间的运行时依赖关系；动态服务依赖图构建器使用部署时服务依赖关系和运行时服务依赖关系，动态地构造服务依赖图；服务依赖图存储器用于存储服务依赖图；

异常传播图构建模块，包括异常传播图构建器、异常传播图存储器，异常传播图构建器使用服务依赖图和输入的异常集，构造异常传播图；异常传播图存储器用于存储异常传播图；

异常根因定位模块，包括异常传播路径构建器、异常根因计算器、异常根因报告器，异常传播路径构建器使用异常传播图，寻找所有可能的异常根因节点，并构建异常的传播路径；异常根因计算器使用异常传播路径，根据路径的个数、路径上边的权重和异常发生的时刻，计算异常的根因分数；异常根因报告器将异常按照分数降序排序，并报告与之对应的异常传播路径。

一种基于动态服务依赖图的异常根因定位方法及系统

技术领域

[0001] 本发明属于智能运维领域，具体涉及一种基于动态服务依赖图的异常根因定位方法。

背景技术

[0002] 微服务架构的系统(简称微服务系统)由大量的细粒度的微服务组成，微服务之间协同工作，实现系统功能，被广泛应用于通信、交通、物流、金融等领域。现代互联网应用，往往使用微服务系统作为底层实现。微服务架构的核心设计思想是，将应用拆分成若干个高内聚，低耦合，功能单一，可独立开发、部署、更新的微服务，以此实现应用的敏捷开发与持续交付。微服务架构使得应用的开发和迭代更加的便捷，但也给系统的运行维护带来的新的挑战，即复杂的微服务系统使得应用容易发生异常且难以诊断异常根因。

[0003] 微服务系统复杂，具体体现在微服务数量庞大，微服务间依赖关系动态变化，微服务的异构实现。由于微服务的功能单一，为了完成系统的功能，微服务系统可能拥有庞大的微服务数量。例如，Netflix的微服务系统运行了6000多个微服务实例，每天需要处理超过20亿个服务请求；Uber的微服务系统也运行了4000多个微服务实例。同时，微服务系统为应对用户需求的变化与应用的发展，可以动态部署与注册微服务实例，不断有新的微服务加入，使得微服务之间的依赖关系动态变化且错综复杂。并且，微服务系统的服务化接口封装，允许系统内不同微服务采取不同的编程语言、服务框架以及通信机制，以应对不同的应用场景。这种系统内的异构性设计与实现，进一步加深了微服务系统的复杂程度。

[0004] 微服务系统的复杂性，导致了其容易发生异常且难以诊断异常的根因。异常是指，系统的运行状态相比于系统的设计期望发生了偏移。微服务的松耦合设计及异构实现，使得微服务间往往通过网络远程调用的形式完成协作，容易产生由网络不稳定、版本不兼容、配置错误、代码缺陷等问题引发的异常。微服务的数量庞大，进一步加深了上述异常产生的可能性。并且，由于服务间依赖关系复杂且动态变化，任何微小的系统异常均可能引起连锁反应，导致多个异常同时发生，其中只有一个或少数几个异常是引发其他异常的根因。这是由于异常会随着服务间依赖关系不断扩散，形成异常传播链，这些因依赖关系而产生的微服务异常之间存在着因果关系，往往异常传播链的源头是异常根因。为了保障系统的可靠性，当检测到系统运行发生异常，运维人员需要第一时间定位异常的根因，并作出相应的处理，如变更的回滚等。否则，异常可能引起服务中断或服务质量下降，并带来严重的损失。国际数据公司曾报告，一小时的服务中断平均会造成余约10万美元的损失。然而，尽管微服务系统可以提供丰富的运维数据，例如日志、关键性能指标、调用链，但差异化的数据格式、稀疏化的有价值信息以及动态变化的服务依赖关系，使得运维人员难以在复杂的微服务系统里定位异常的根因，严重时导致异常不能及时修复，并给企业带来更严重的经济损害与声誉损害。

[0005] 然而，现有微服务系统的异常根因定位方法，仍然存在不足，往往忽略了服务依赖的动态变化以及根因定位的可解释性。现有方法，大多依赖于服务依赖图。服务依赖图用于

刻画服务之间的依赖关系，这种依赖关系也可以用于刻画服务之间的异常传播，从而有助于运维人员进行异常根因定位。基于服务依赖图的异常根因定位首先通过指标、系统日志或追踪数据构造服务依赖图，然后当异常发生时，从服务依赖图中的异常节点出发，通过图搜索、随机游走等算法得到引起异常的候选异常根因集合，然后通过异常分数、与异常节点的相关性或被访问次数等方式对候选异常根因进行排序。然而，现有方法，往往依赖于静态服务依赖图，忽略了服务依赖之间的动态变化，导致构建的服务依赖与真实情况存在差异，限制了现有异常根因定位方法的准确性。与此同时，现有方法，往往只提供了异常的根因服务或根因指标，缺乏可解释性，导致运维人员难以快速地理解报告结果及判断根因的准确性。

发明内容

[0006] 为了减少异常根因定位不准确导致的额外分析与回滚，降低运维成本，提高异常定位的可解释性，保障微服务系统的可靠性，本发明提供了一种基于动态服务依赖图的异常根因定位方法。基于服务配置信息提取部署时服务依赖关系，基于服务间关键性能指标的相关性及日志频率信息，发现运行时服务依赖关系，动态地构建服务依赖图；基于输入的异常服务及服务依赖图，自动构建异常传播图；基于深度优先搜索构建异常传播路径，并计算异常的根因分数，定位异常根因服务，并报告异常传播路径。

[0007] 本发明中的动态服务依赖图构建、异常传播图构建、异常根因定位都是自动进行，无需人工参与，节省了人力成本。本发明的异常根因定位，基于动态服务依赖图，可以更好地捕捉运行时服务依赖关系的变化，有助于刻画更加精确的异常传播关系，提升了异常根因定位的能力；适用于各种类型异常的根因定位，具有很好的通用性；直接定位到异常的服务，并提供异常的传播路径，具有很好的实用性和可解释性。

[0008] 本发明提供的技术方案是：

[0009] 一种基于动态服务依赖图的异常根因定位方法，其特征在于，包括动态服务依赖图构建、异常传播图构建、异常根因定位；具体步骤包括：

[0010] 1)动态服务依赖图构建，具体执行如下步骤：

[0011] 11)提取部署时服务依赖：此处的原理是，部署在同一位置(例如，主机)上或者依赖于同一资源(例如，数据库)的两个微服务，存在部署时依赖关系。从配置管理数据库(Configuration Management Database,CMDB)中提取服务的配置信息，包括部署位置信息和关联资源信息。定义待收集配置信息的属性集合$K=(k_1,k_2,...,k_n)$，使得收集到的配置信息$c_i=(k_i,v_i)$必有$k_i \in K$，其中，k_i表示配置信息的属性(元数据)，v_i表示配置信息的值。记一个服务s_i的配置集合为$C_i=(c_1,c_2,...,c_n)$，若服务s_i与服务s_j的配置集合存在交集，即$C_i \cap C_j \neq \Phi$，则服务s_i与服务s_j之间存在部署依赖。

[0012] 12)提取运行时服务依赖：此处的原理是，若随着系统负载的变化，两服务产生的日志数量或关键性能指标(Key Performance Indicator,KPI)存在因果关系，则两服务存在运行时依赖，且因果关系的强弱可以表示运行时依赖关系的强弱。此处使用的因果推断算法是PC算法，原因是在给定可靠的条件独立性检验方法的情况下，PC算法可以处理各种类型的数据分布和因果关系，相比于其他因果推断方法具有复杂低、效果好的优势。PC算法通过条件独立性检验判断变量之间的因果关系，再利用d-分离条件确定因果关系之间的方

向。此处使用的条件独立性检验方法是G^2条件交叉熵度量，如式(1)所示，其服从自由度为D的x^2分布，如式(2)所示。

[0013] $$G^2 = 2mCE(X,Y|Z) = \sum_z P(z) \sum_x \sum_y P(x,y|z)\log\left(\frac{P(x,y|z)}{p(x|z)\cdot p(y|z)}\right)$$ 式（1）

[0014] $D=(N_X-1)(N_Y-1)\prod_{Z'\in Z}N_{Z'}$ 式(2)

[0015] 其中，$X\epsilon R^{N_X}$，$Y\epsilon R^{N_Y}$，$Z'\epsilon R^{N_{Z'}}$，Z是Z′的集合，m是采样的个数。

[0016] 13)构建动态服务依赖图：每次执行异常根因定位任务时，均会执行步骤11)和步骤12)，动态地获取部署时服务依赖和运行时服务依赖。并基于部署时服务依赖和运行时服务依赖，构建服务依赖图G＝<V,E>，其中G是有向图(Directed Graph,DG)，V表示服务，是G的节点；E表示依赖关系，是G的边。构造规则为，若服务s_i与服务s_j之间存在运行时依赖，则添加s_i与s_j之间的有向边，边的权重为s_i与s_j之间的G^2值。若服务s_i与服务s_j之间存在部署时依赖，则添加s_i到s_j及s_j到s_i的有向边，边的权重均为运行时依赖边权的均值。

[0017] 2)异常传播图构建，具体执行如下步骤：

[0018] 21)构建异常传播图：将输入的异常集合A＝{a_1,a_2,...,a_n}标记在服务依赖图G上，将标为异常的服务节点取出，形成G的子图G′，同时保留节点间的边，节点的值为异常发生时刻的倒数。其中，a_i＝(s_i,t_i)表示第i个异常，s_i表示发生异常a_i的服务，t_i表示发生异常a_i的时刻。为了方便计算，将异常发生的时刻进行标准化。设最先发生异常的时刻为1；其余异常按照发生的顺序，发生时刻在1的基础上递增。

[0019] 3)异常根因定位，具体执行如下步骤：

[0020] 31)构建异常传播路径：在异常传播图G′上，随机选择一个异常的服务s'_0出发，利用深度优先搜索，寻找候选根因节点及其异常传播路径。

[0021] 32)计算异常根因分数：对于s'_0，利用异常传播图G′及异常传播路径Path(s'_0,s'_t)，计算每一个$s'_t\in$R的异常根因分数。计算公式如式(3)所示。

[0022] $$score(s'_t) = \sum_{k=1}^{M}\prod_{i=1}^{N} w_{k,i} * s_i$$ 式（3）

[0023] 其中，score(s'_t)表示异常服务s'_t的异常根因分数，M表示s'_0到s'_t的异常传播路径个数，N表示传播路径o_k中的跳数，$w_{k,i}$表示第k条传播路径中节点i与节点i-1之间的依赖权重，s_i表示节点i的值，即异常发生时刻的倒数。此处根因分数计算公式设计的原理是，越早发生的异常越可能是根因，和有越多条传播路径的异常越可能是根因。

[0024] 33)报告异常根因及传播路径：对异常根因按照分数进行降序排列，并报告与之对应的异常传播路径。

[0025] 本发明进一步提供一种基于动态服务依赖图的异常根因定位系统，其特征在于，包括动态服务依赖图构建模块、异常传播图构建模块、异常根因定位模块。

[0026] 动态服务依赖图构建模块，包括部署时服务依赖关系发现器、运行时服务依赖关系发现器、动态服务依赖图构建器、服务依赖图存储器；部署时服务依赖关系发现器使用服务配置信息，发现微服务之间的部署时依赖关系；运行时服务依赖关系发现器使用服务运行日志及关键性能指标，发现微服务之间的运行时依赖关系；动态服务依赖图构建器使用部署时服务依赖关系和运行时服务依赖关系，动态地构造服务依赖图；服务依赖图存储器用于存储服务依赖图；

[0027] 异常传播图构建模块，包括异常传播图构建器、异常传播图存储器。异常传播图构

建器使用服务依赖图和输入的异常集，构造异常传播图；异常传播图存储器用于存储异常传播图；

[0028] 异常根因定位模块，包括异常传播路径构建器、异常根因计算器、异常根因报告器。异常传播路径构建器使用异常传播图，寻找所有可能的异常根因节点，并构建异常的传播路径；异常根因计算器使用异常传播路径，根据路径的个数、路径上边的权重和异常发生的时刻，计算异常的根因分数；异常根因报告器将异常按照分数降序排序，并报告与之对应的异常传播路径。

[0029] 与现有技术相比，本发明的有益效果是：

[0030] 本发明提供了一种基于动态服务依赖图的异常根因定位方法及系统通过读取服务配置信息以及服务运行日志和关键性能指标，动态地构建服务依赖图。并将服务依赖图进行存储，用于后续生成异常传播图。当系统发生异常后，根据服务依赖图和异常集合，构建异常传播图。然后利用深度优先搜索，遍历所有可能的根因异常及其传播路径，利用异常传播路径的个数、路径上边的权重和异常发生的时刻，计算异常的根因分数，按照分数降序排序，并报告与之对应的异常传播路径。本发明能够实现自动构建动态服务依赖图，生成异常传播图，并构建异常传播路径，计算根因分数，以此定位异常根因，并报告相应的传播路径。本发明主要具有以下特点：

[0031] (一)本发明提供的系统和方法以服务配置信息和运行时信息(关键性能指标和日志)为基础，自动构建动态服务依赖图。

[0032] (二)本发明提供的系统和方法能够使用输入的异常和服务依赖图，自动构建异常传播图。

[0033] (三)本发明提供的系统和方法能够定位异常根因服务的同时，提供异常传播路径，具有很好的实用性和可解释性。

[0034] 利用本发明的技术方案，可以实现自动构建动态服务依赖图，构建异常传播图，定位异常根因，并提供异常的传播路径，适用于微服务系统。

附图说明

[0035] 图1是本发明提供的基于动态服务依赖图的异常根因定位方法；

[0036] 图2是本发明提供的基于动态服务依赖图的异常根因定位系统。

具体实施方式

[0037] 下面结合附图，通过实施例进一步描述本发明，但不以任何方式限制本发明的范围。

[0038] 图1是本发明提供的基于动态服务依赖图的异常根因定位方法的流程框图。本发明包括动态服务依赖图构建、异常传播图构建、异常根因定位；

[0039] 动态服务依赖图构建使用服务配置信息以及服务运行日志和关键性能指标，挖掘服务间的部署时依赖关系和运行时依赖关系，动态地构造服务依赖图。配置信息用于描述服务所处的位置，及其所依赖的资源与服务。每个配置项以两元组的形式存在，包括配置属性和配置值。运行日志用于记录系统的运行状况，包括关键变量的输出以及关键运行位置的标记等。运行日志以时间序列文本的形式存在，在本发明中被转换为了日志频率的时间

序列的形式。关键性能指标用于监控系统的运行状态，用于监控系统是否发生异常。关键性能指标以时间序列的形式存在。服务依赖图是有向图，刻画了微服务系统中各个服务之间的依赖关系，图的节点是服务，边是服务之间的依赖。服务依赖图随着系统运行数据的积累，动态构建，具有对系统迭代的自适应性。

[0040] 异常传播图构建使用输入的异常集合和服务依赖图，构建异常传播图，用于之后的传播路径分析。每个异常项以两元组的形式存在，包括发生异常的服务和发生异常的时刻。异常传播图也是有向图的形式，是服务依赖图的子图。

[0041] 异常根因定位使用异常传播图，寻找所有可能的异常根因节点，并构建与之对应的异常传播路径，用于计算节点的根因分数，并降序排序生成异常根因报告。异常根因报告包含根因服务分数，及其对应的异常传播路径。

[0042] 针对上述基于动态服务依赖图的异常根因定位方法，所述动态服务依赖图构建具体执行如下步骤：

[0043] 11)提取部署时服务依赖：此处的原理是，部署在同一位置(例如，主机)上或者依赖于同一资源(例如，数据库)的两个微服务，存在部署时依赖关系。从配置管理数据库(Configuration Management Database,CMDB)中提取服务的配置信息，包括部署位置信息和关联资源信息。定义待收集配置信息的属性集合$K=(k_1,k_2,...,k_n)$，使得收集到的配置信息$c_i=(k_i,v_i)$必有$k_i \in K$，其中，k_i表示配置信息的属性(元数据)，v_i表示配置信息的值。记一个服务s_i的配置集合为$C_i=(c_1,c_2,...,c_n)$，若服务s_i与服务s_j的配置集合存在交集，即$C_i \cap C_j \neq \Phi$，则服务s_i与服务s_j之间存在部署依赖。

[0044] 12)提取运行时服务依赖：此处的原理是，若着系统负载的变化，两服务产生的日志数量或关键性能指标(Key Performance Indicator,KPI)存在因果关系，则两服务存在运行时依赖，且因果关系的强弱可以表示运行时依赖关系的强弱。此处使用的因果推断算法是PC算法，原因是在给定可靠的条件独立性检验方法的情况下，PC算法可以处理各种类型的数据分布和因果关系，相比于其他因果推断方法具有复杂低、效果好的优势。PC算法通过条件独立性检验判断变量之间的因果关系，再利用d-分离条件确定因果关系之间的方向。此处使用的条件独立性检验方法是G^2条件交叉熵度量，如式(1)所示，其服从自由度为D的x^2分布，如式(2)所示。

[0045] $$G^2 = 2mCE(X,Y|Z) = \sum_z P(z) \sum_x \sum_y P(x,y|z) \log \left(\frac{P(x,y|z)}{p(x|z) \cdot p(y|z)}\right) \quad \text{式（1）}$$

[0046] $$D=(N_X-1)(N_Y-1)\prod_{Z' \in Z} N_{Z'} \quad \text{式(2)}$$

[0047] 其中，$X \epsilon R^{Nx}$，$Y \epsilon R^{Ny}$，$Z' \epsilon R^{Nz'}$，Z是Z′的集合，m是采样的个数。

[0048] 121)确定运行时服务依赖关系权重：具体来说，此处使用的KPI是服务级别目标(Service Level Objective,SLO)指标，如服务请求延迟，用于评估一个服务是否运行正常。记一个服务s_i的KPI时间序列为$T_i=\{t_1,t_2,...,t_n\}$，日志频数序列为$L_i=\{l_1,l_2,...,l_n\}$，时间窗口大小为φ，时间窗口个的总数为n。初始时，假设在m个服务中，任意两个服务之间存在因果关系。利用时间序列数据T，计算任意s_i与s_j之间的G^2值，并查询其在x^2分布的p值，若p值＞ξ，则s_i与s_j之间的条件独立性假设被接受，否则拒绝。若s_i与s_j之间的条件独立性假设被接受，则记录此时条件Z作为s_i与s_j的分割条件$S(s_i,s_j)$。同理，利用日志频数序列数据L，对s_i与s_j进行条件独立性检验。如果针对s_i与s_j的关于T和L的两次独立性假设均被

接受，则判定sa_i与s_j之间没有因果关系，否则判定s_i与s_j之间存在因果关系。遍历所有的s_i与s_j对，直至确定所有的s_i与s_j对之间的因果关系。

[0049] 122)确定运行时服务依赖关系方向：之后用d-分离条件确定因果关系之间的方向。d-分离条件共有四条规则，

[0050] (1)对于任意不相邻(没有因果关系)的两个变量X和Y，且拥有共同的邻居变量Z，若$Z \notin S(X,Y)$，则将X-Z-Y赋予方向X→Z←Y。

[0051] (2)若存在X→Y，则将所有Y-Z赋予方向Y→Z。

[0052] (3)若存在X→Z→Y，则将所有X-Y赋予方向X→Y。

[0053] (4)若同时存在X-Z_1→Y和X-Z_2→Y，则将所有X-Y赋予方向X→Y。

[0054] 其中规则(1)优先于，规则(2)(3)(4)，即确保所有的规则(1)都执行后，再执行规则(2)(3)(4)。规则(2)(3)(4)的执行没有先后顺序。

[0055] 若执行完上述规则后，仍无法确定s_i与s_j之间依赖的方向，则添加双向依赖，即添加s_i到s_j的依赖关系，同时添加s_j到s_i的依赖关系。

[0056] 13)构建动态服务依赖图：每次执行异常根因定位任务时，均会执行步骤11)和步骤12)，动态地获取部署时服务依赖和运行时服务依赖。并基于部署时服务依赖和运行时服务依赖，构建服务依赖图G＝<V,E>，其中G是有向图(Directed Graph,DG)，V表示服务，是G的节点；E表示依赖关系，是G的边。构造规则为，若服务s_i与服务s_j之间存在运行时依赖，则添加s_i与s_j之间的有向边，边的权重为s_i与s_j之间的G^2值。若服务s_i与服务s_j之间存在部署时依赖，则添加s_i到s_j及s_j到s_i的有向边，边的权重均为运行时依赖边权的均值。

[0057] 针对上述基于动态服务依赖图的异常根因定位方法，所述异常传播图构建具体执行如下步骤：

[0058] 21)构建异常传播图：将输入的异常集合A＝{a_1,a_2,...,a_n}标记在服务依赖图G上，将标为异常的服务节点取出，形成G的子图G′，同时保留节点间的边，节点的值为异常发生时刻的倒数。其中，a_i＝(s_i,t_i)表示第i个异常，s_i表示发生异常a_i的服务，t_i表示发生异常a_i的时刻。为了方便计算，将异常发生的时刻进行标准化。设最先发生异常的时刻为1；其余异常按照发生的顺序，发生时刻在1的基础上递增。

[0059] 针对上述基于动态服务依赖图的异常根因定位方法，所述异常根因定位具体执行如下步骤：

[0060] 31)构建异常传播路径：在异常传播图G′上，随机选择一个异常的服务s'_0出发，利用深度优先搜索，寻找候选根因节点及其异常传播路径。

[0061] 311)寻找候选根因节点集合：在寻找候选根因节点阶段，初始化，s'_i＝s'_0，o_k＝Φ，R＝Φ，然后执行如下递归步骤：

[0062] (1)若$s'_i \in o_k$，则返回上级调用。

[0063] (2)将s'_i加入到o_k。

[0064] (3)若s'_i不存在相邻的异常节点，则将s'_i加入到候选根因节点集合R中。

[0065] (4)若s'_i存在相邻的异常节点s'_j，对每一个s'_j，若$s'_j \notin o_k$，令s'_i＝s'_j，并执行步骤(1)。

[0066] (5)返回上级调用。

[0067] 递归执行步骤(1)到步骤(5)，直到不再有新的异常服务加入到R中。

[0068] 312)确定异常传播路径:在寻找传播路径阶段,对于每一个$s'_t \in R$,记从异常服务s'_0到异常服务s'_t的异常传播路径集为$Path(s'_0,s'_t)=\{o_1,o_2,...,o_n\}$,其中$o_k$为一条异常传播路径,$o_k=\{s'_0,...,s_t'\}$。初始化,$s'_i=s'_0$,$o_k=\Phi$,$Path(s'_0,s'_t)=\Phi$,然后执行如下递归步骤

[0069] (1)若$s'_i \in o_k$,则返回上级调用。

[0070] (2)将s'_i加入到o_k。

[0071] (3)若$s'_i==s'_t$,则将o_k加入到$Path(s'_0,s'_t)$。

[0072] (4)若s'_i存在相邻的异常服务节点s'_j,对每一个s'_j,若$s'_j \notin o_k$,令$s'_i=s'_j$,并执行步骤(1)。

[0073] (5)返回上级调用。

[0074] 递归执行步骤(1)到步骤(5),直到不再有新的异常传播路径加入到$Path(s'_0,s'_t)$中。

[0075] 32)计算异常根因分数:对于s'_0,利用异常传播图G'及异常传播路径$Path(s'_0,s'_t)$,计算每一个$s'_t \in R$的异常根因分数。计算公式如式(3)所示。

[0076] $$score(s'_t)=\sum_{k=1}^{M}\prod_{i=1}^{N}w_{k,i}*s_i \qquad 式(3)$$

[0077] 其中,$score(s'_t)$表示异常服务s'_t的异常根因分数,M表示s'_0到s'_t的异常传播路径个数,N表示传播路径o_k中的跳数,$w_{k,i}$表示第k条传播路径中节点i与节点i-1之间的依赖权重,s_i表示节点i的值,即异常发生时刻的倒数。此处根因分数计算公式设计的原理是,越早发生的异常越可能是根因,和有越多条传播路径的异常越可能是根因。

[0078] 33)报告异常根因及传播路径:对异常根因按照分数进行降序排列,并报告与之对应的异常传播路径。

[0079] 图2是本发明提供的基于动态服务依赖图的异常根因定位系统的结构框图。

[0080] 本发明提供了一种实现基于动态服务依赖图的异常根因定位方法的系统,系统以配置信息、运行日志、关键性能指标、异常集合作为输入,包括动态服务依赖图构建模块、异常传播图构建模块、异常根因定位模块;

[0081] 下面分别对不同的模块进行具体说明。

[0082] S1)动态服务依赖图构建模块

[0083] 动态服务依赖图构建模块的功能是,基于服务配置信息、运行日志、关键性能指标,构建动态服务依赖图。该模块包含四个子模块:

[0084] S11)部署时服务依赖关系发现器

[0085] 部署时服务依赖关系发现器基于服务配置信息,挖掘部署在同一位置或依赖同一资源的微服务之间的部署时依赖关系。

[0086] S12)运行时服务依赖关系发现器

[0087] 运行时服务依赖关系发现器基于服务运行日志和关键性能指标,挖掘微服务之间的运行时依赖关系。

[0088] S13)动态服务依赖图构建器

[0089] 动态服务依赖图构建器基于服务部署时依赖关系和运行时依赖关系,动态地构建服务依赖图。

[0090] S14)服务依赖图存储器

[0091] 服务依赖图存储器以矩阵的形式存储服务依赖图,并提供对服务依赖图高性能的查询。

[0092] S2)异常传播图构建模块

[0093] 异常传播图构建模块的功能是根据服务依赖图和输入的异常集合,构建异常传播图。该模块包含两个子模块:

[0094] S21)异常传播图构建器

[0095] 异常传播图构建器根据服务依赖图和输入的异常集合,在服务依赖图上标记异常并生成子图,构造异常传播图。

[0096] S22)异常传播图存储器

[0097] 异常传播图存储器以矩阵的形式存储异常传播图,并提供对异常传播图高性能的查询。

[0098] S3)异常根因定位模块

[0099] 异常根因定位模块的功能是根据异常传播图,寻找所有可能的异常根因节点,并构建其异常传播路径,计算根因分数,生成异常根因报告。该模块分为三个子模块:

[0100] S31)异常传播路径构建器

[0101] 异常传播路径构建器使用异常传播图,寻找所有可能的异常根因节点,并构建异常的传播路径。

[0102] S32)异常根因计算器

[0103] 异常根因计算器使用异常传播路径,根据异常传播路径的个数、路径上边的权重和异常发生的时刻,计算异常的根因分数。

[0104] S32)异常根因报告器

[0105] 异常根因报告器将异常按照分数降序排序,并报告与之对应的异常传播路径。

[0106] 需要注意的是,公布实施例的目的在于帮助进一步理解本发明,但是本领域的技术人员可以理解:在不脱离本发明及所附权利要求的精神和范围内,各种替换和修改都是可能的。因此,本发明不应局限于实施例所公开的内容,本发明要求保护的范围以权利要求书界定的范围为准。

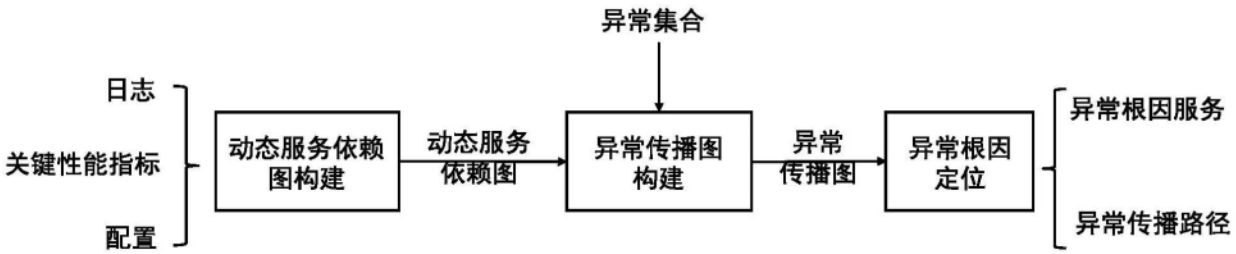

异常集合
日志
关键性能指标
配置
动态服务依赖图构建
动态服务依赖图
异常传播图构建
异常传播图
异常根因定位
异常根因服务
异常传播路径

图1

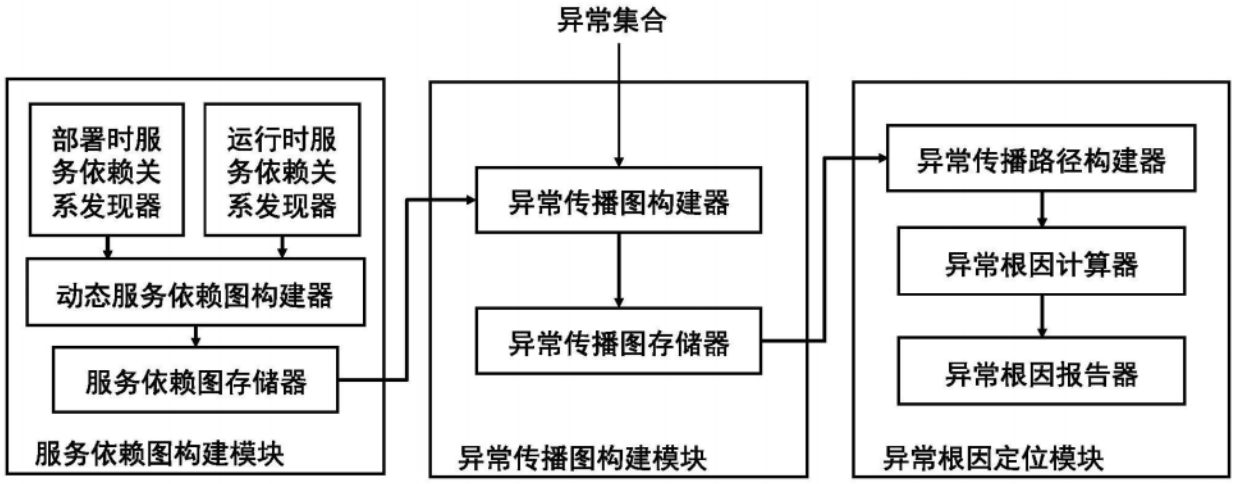

异常集合
部署时服务依赖关系发现器
运行时服务依赖关系发现器
动态服务依赖图构建器
服务依赖图存储器
服务依赖图构建模块
异常传播图构建器
异常传播图存储器
异常传播图构建模块
异常传播路径构建器
异常根因计算器
异常根因报告器
异常根因定位模块

图2