(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 116569179 A
(43)申请公布日 2023.08.08

(21)申请号 202180078587.6
(22)申请日 2021.10.21
(30)优先权数据
17/106,298 2020.11.30 US
(85)PCT国际申请进入国家阶段日
2023.05.23
(86)PCT国际申请的申请数据
PCT/CN2021/125261 2021.10.21
(87)PCT国际申请的公布数据
WO2022/111154 EN 2022.06.02
(71)申请人 国际商业机器公司
地址 美国纽约
(72)发明人 康辉 阙欣宇 邓豫
S•古文卡亚 B•达莫拉
(74)专利代理机构 中国贸促会专利商标事务所有限公司 11038
专利代理师 程晨
(51)Int.Cl.
G06N 3/044(2023.01)
G06N 3/08(2023.01)

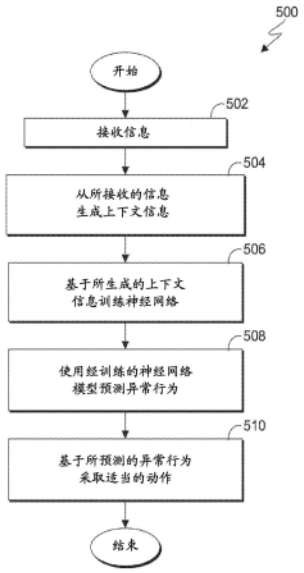
权利要求书2页 说明书17页 附图8页

(54)发明名称

主动异常检测

(57)摘要

提供了计算机实现的方法、计算机程序产品和计算机系统。例如,方法可响应于接收到请求而收集针对微服务应用的正常行为的请求序列的跟踪数据和规范。方法可随后从所收集的跟踪数据和规范生成请求上下文特征。方法随后可基于所生成的上下文特征来训练神经网络模型;以及使用经训练的神经网络模型来预测微服务应用的异常行为。



CN 116569179 A

1.一种计算机实现方法,包括:

响应于接收到请求,收集针对微服务应用的正常行为的请求序列的跟踪数据和规范;

从所收集的跟踪数据和规范生成请求上下文特征;

基于所生成的上下文特征来训练神经网络模型;以及

使用经训练的神经网络模型来预测所述微服务应用的异常行为。

2.根据权利要求1所述的计算机实现的方法,进一步包括:

生成与所预测的异常行为相关联的可视化。

3.根据权利要求1所述的计算机实现的方法,进一步包括:

生成所预测的异常行为的根本原因报告。

4.根据权利要求1所述的计算机实现的方法,进一步包括:

提供所预测的异常行为的系统模拟。

5.根据权利要求1所述的计算机实现的方法,其中,所述跟踪数据提供将日志分离成单独的请求的层级数据结构。

6.根据权利要求1所述的计算机实现的方法,其中,所述神经网络模型是递归神经网络。

7.根据权利要求1所述的计算机实现的方法,其中,所述请求上下文特征包括:

数据结构,其包括请求的三级信息:请求规范、微服务路径和功能路径。

8.根据权利要求1所述的计算机实现的方法,其中,从所收集的跟踪数据和规范生成请求上下文特征包括:

集成与所述请求相关联的请求间因子和请求内因子。

9.一种计算机程序产品,包括:

一个或多个计算机可读存储介质和存储在所述一个或多个计算机可读存储介质上的程序指令,所述程序指令包括:

用于响应于接收到请求而收集针对微服务应用的正常行为的请求序列的跟踪数据和规范的程序指令;

用于从所收集的跟踪数据和规范生成请求上下文特征的程序指令;

用于基于所生成的上下文特征来训练神经网络模型的程序指令;以及

用于使用经训练的神经网络模型来预测所述微服务应用的异常行为的程序指令。

10.根据权利要求9所述的计算机程序产品,其中,存储在所述一个或多个计算机可读存储介质上的所述程序指令进一步包括:

用于生成与所预测的异常行为相关联的可视化的程序指令。

11.根据权利要求9所述的计算机程序产品,其中,存储在所述一个或多个计算机可读存储介质上的所述程序指令进一步包括:

用于生成所预测的异常行为的根本原因报告的程序指令。

12.根据权利要求9所述的计算机程序产品,其中,存储在所述一个或多个计算机可读存储介质上的所述程序指令进一步包括:

用于提供所预测的异常行为的系统模拟的程序指令。

13.根据权利要求9所述的计算机程序产品,其中,所述跟踪数据提供将日志分离成单独的请求的层级数据结构。

14.根据权利要求9所述的计算机程序产品,其中,所述神经网络模型是递归神经网络。

15.根据权利要求9所述的计算机程序产品,其中,所述请求上下文特征包括:

数据结构,其包括请求的三级信息:请求规范、微服务路径和功能路径。

16.根据权利要求9所述的计算机程序产品,其中,用于从所收集的跟踪数据和规范生成请求上下文特征的所述程序指令包括:

用于集成与所述请求相关联的请求间因子和请求内因子的程序指令。

17.一种计算机系统,包括:

一个或多个计算机处理器;

一个或多个计算机可读存储介质;以及

存储在所述一个或多个计算机可读存储介质上的程序指令,用于由所述一个或多个计算机处理器中的至少一个执行,所述程序指令包括:

用于响应于接收到请求而收集针对微服务应用的正常行为的请求序列的跟踪数据和规范的程序指令;

用于从所收集的跟踪数据和规范生成请求上下文特征的程序指令;

用于基于所生成的上下文特征来训练神经网络模型的程序指令;以及

用于使用经训练的神经网络模型来预测所述微服务应用的异常行为的程序指令。

18.根据权利要求17所述的计算机系统,其中,存储在所述一个或多个计算机可读存储介质上的所述程序指令进一步包括:

用于生成与所预测的异常行为相关联的可视化的程序指令。

19.根据权利要求17所述的计算机系统,其中,存储在所述一个或多个计算机可读存储介质上的所述程序指令进一步包括:

用于生成所预测的异常行为的根本原因报告的程序指令。

20.根据权利要求17所述的计算机系统,其中,存储在所述一个或多个计算机可读存储介质上的所述程序指令进一步包括:

用于提供所预测的异常行为的系统模拟的程序指令。

主动异常检测

背景技术

[0001] 本发明总体上涉及主动异常检测，并且具体涉及使用请求上下文数据和神经网络的微服务应用的主动异常检测。

[0002] 微服务架构将应用安排为松散耦合的服务的集合。微服务不是单体式应用(例如，网络控制器或服务于前端的后端)内的层。以此方式，微服务架构本身适用于连续的交付软件开发过程。对应用的小部分的改变仅需要重建和重新部署仅一个或少量服务。

[0003] 通常，微服务架构可以用于云原生应用、无服务器计算、以及使用轻量级容器部署的应用。在单体式方法中，支持三个功能(例如，框架、数据库、消息代理等)的应用将不得不整体缩放，即使这些功能中仅有一个具有资源约束。利用微服务，只需要将支持具有资源约束的功能的微服务进行扩展，从而提供资源和成本优化的有益效果。

[0004] 机器学习(ML)是不使用显式指令而依赖于模式和推断来供计算机系统用来执行特定任而的算法和统计模型的科学研究。机器学习被视为人工智能的子集。机器学习算法基于样本数据(称为训练数据)建立数学模型，以便作出预测或决定，而不被显式地编程为执行任务。机器学习算法被用于各种各样的应用中，诸如电子邮件过滤和计算机视觉，其中开发用于有效地执行任务的常规算法是困难的或不可行的。

[0005] 在机器学习中，超参数是在模型外部的配置，并且其值不能从数据中估计。在过程中使用超参数以帮助估计模型参数。相比之下，在学习(例如，训练)过程开始之前设置超参数，经由训练导出其他参数的值。不同的模型训练算法需要不同的超参数，一些简单的算法(例如最小二乘回归)不需要超参数。给定超参数集合，训练算法例如从数据学习参数值，最小绝对收缩和选择算子(LASSO)是将正则化超参数添加到最小二乘回归的算法，需要在通过训练算法估计参数之前设置。类似的机器学习模型可能需要不同的超参数(例如，不同的约束、权重或学习速率)来概括不同的数据模式。

[0006] 深度学习是基于一组算法的机器学习的分支，所述一组算法通过使用具有复杂的结构或以其他方式通常由多个非线性变换组成的模型架构对数据中的高级抽象进行建模。深度学习是基于学习数据的表示的更广泛的机器学习方法族的一部分。观察(例如，图像)可以以许多方式来表示，诸如每个像素的强度值的向量，或者以更抽象的方式表示为边、特定形状的区域等的集合。一些表示使得更容易从实例学习任务(例如，面部识别或面部表情识别)。深度学习算法通常使用非线性处理单元的许多层的级联来进行特征提取和变换。每一连续层使用来自先前层的输出作为输入。算法可以是监督的或非监督的，并且应用包括模式分析(非监督的)和分类(监督的)。深度学习模型包括从生物系统中的信息处理和分布式通信节点启发的人工神经网络(ANN)。ANN与生物脑具有各种差异。

[0007] 神经网络(NN)是由生物神经网络启发的计算系统。NN不仅仅是算法，而是用于许多不同机器学习算法一起工作和处理复杂数据输入的框架。通过考虑实例，这种系统“学习”执行任务，通常不使用任何任务特定的规则进行编程。例如，在图像识别中，NN通过分析正确地标记为“猫”或“非猫”的实例图像来学习识别包含猫的图像，并且使用结果来识别其他图像中的猫。NN在没有关于猫的任何先验知识(例如猫具有毛皮、尾巴、胡须和尖耳)的情

况下实现这一点。相反，NN从学习材料自动生成识别特性。NN基于称为人工神经元的连接单元或节点的集合，其松散地对生物脑中的神经元建模。如同生物脑中的突触，每一连接可将信号从一个人工神经元传送至另一人工神经元。接收信号的人工神经元可以处理该信号并且然后将该信号传递到附加的人工神经元。

[0008] 在公共NN实现中，人工神经元之间的连接处的信号是实数，并且每个人工神经元的输出由其输入之和的某个非线性函数来计算。人工神经元之间的连接被称为“边”。人工神经元和边通常具有随学习进展而调整的权重。权重增大或减小连接处的信号的强度。人工神经元可具有阈值，使得仅在聚集信号跨越该阈值时才发送该信号。典型地，人工神经元被聚集成层。不同层可对其输入执行不同种类的变换。信号从第一层(输入层)行进到最后一层(输出层)，这可能发生在经过这些层多次之后。

发明内容

[0009] 根据本发明的方面，提供了一种计算机实现的方法。所述方法包括：响应于接收到请求，收集针对微服务应用的正常行为的请求序列的跟踪数据和规范；从所收集的跟踪数据和规范生成请求上下文特征；基于所生成的上下文特征来训练神经网络模型；以及使用经训练的神经网络模型来预测所述微服务应用的异常行为。

附图说明

[0010] 现在将参考以下附图仅通过实例的方式来描述本发明的优选实施例，在附图中：

[0011] 图1描绘了根据本发明的实施例的计算环境的框图；

[0012] 图2描绘了根据本发明的实施例的用于微服务的异常检测器的示例框图；

[0013] 图3描绘了根据本发明的实施例的神经网络模型的设计的示例框图；

[0014] 图4描绘了根据本发明的实施例的捕获针对个体请求的请求内因子的神经网络模型的示例框图；

[0015] 图5示出了根据本发明的实施例的用于预测异常行为的操作步骤；

[0016] 图6示出了根据本发明的实施例的示例图；

[0017] 图7A和7B示出了根据本发明的实施方式的示例数据收集代码；以及

[0018] 图8是根据本发明的实施例的示例系统的框图。

具体实施方式

[0019] 因为松散耦合的组件提供更好的可扩展性、灵活性、可维护性和加速的开发者生产率，所以微服务架构通常用于在混合云环境中部署的应用的本发明的实施例。这样的应用由许多服务组成，这些服务进而被复制到若干实例并且在不同地理位置上运行。随着时间的推移，由于异常引起的性能下降可能发生。照此，本发明的实施例还认识到，检测微服务应用中的异常是关键任务，其使得能够采取可有助于减轻停机时间和生产率损失的某些动作。由于有限的可观察性，当前的系统与监视微服务应用和优化性能相斗争。进一步，本发明的实施例认识到，异常检测的典型方法当前缺乏考虑服务之间的空间和时间依赖性的能力，这可导致更多的误报。因而，本发明的实施例提供了用于改进当前异常检测系统的解决方案，并为技术服务支持人员提供了用于管理复杂微服务应用的高效工具。例如，本发明

的实施例使用神经网络基于上下文数据来检测异常。以此方式，如稍后在本说明书中更详细描述的，本发明的各实施例使用神经网络方法，联合地考虑请求上下文数据中可用的依赖性，预测应用中的性能异常(例如，服务水平协议(SLA)违反)。本发明的实施例然后可以产生通知，且随后在用户意识到之前校正检测到的异常。

[0020] 图1是示出根据本发明的一个实施例的计算环境(一般指定为计算环境100)的功能框图。图1仅提供一个实现方式的图示并且不暗示关于其中可以实现不同实施例的环境的任何限制。本领域技术人员可对所描述的环境作出许多修改，而不脱离权利要求书所述的本发明的范围。

[0021] 计算环境100包括客户端计算设备102和服务器计算机108，其全部通过网络106互连。客户端计算设备102和服务器计算机108可以是独立的计算机设备、管理服务器、web服务器、移动计算设备或能够接收、发送和处理数据的任何其他电子设备或计算系统。在其他实施例中，客户端计算设备102和服务器计算机108可表示诸如在云计算环境中利用多个计算机作为服务器系统的服务器计算系统。在另一实施例中，客户端计算设备102和服务器计算机108可以是膝上型计算机、平板计算机、上网本计算机、个人计算机(PC)、台式计算机、个人数字助理(PDA)、智能电话、或能够与计算环境100内的各种组件和其他计算设备(未示出)通信的任何可编程电子设备。在另一实施例中，客户端计算设备102和服务器计算机108各自表示利用集群计算机和组件(例如，数据库服务器计算机、应用服务器计算机等)的计算系统，所述集群计算机和组件在计算环境100内被访问时充当单个无缝资源池。在一些实施例中，客户端计算设备102和服务器计算机108是单个设备。客户端计算设备102和服务器计算机108可包括能够执行机器可读程序指令的内部和外部硬件组件，如关于图6更详细描绘和描述的。

[0022] 在该实施例中，客户端计算设备102是与用户相关联的用户设备并且包括应用104。应用104与服务器计算机108通信以(例如，使用TCP/IP)访问异常检测器110或接收服务请求和数据库信息。应用104还可与异常检测器110通信以标识与接收到的请求相关联的上下文特征，生成或以其他方式训练神经网络模型，并使用所生成的神经网络模型来预测在微服务应用内处理的未来请求，如参见图2-5更详细地讨论的。

[0023] 网络106可以是例如电信网络、局域网(LAN)、广域网(WAN)(诸如互联网)或三者的组合，并且可以包括有线、无线或光纤连接。网络106可以包括能够接收和传送数据、语音和/或视频信号(包括包含语音、数据和视频信息的多媒体信号)的一个或多个有线和/或无线网络。一般而言，网络106可以是将支持客户端计算设备102和服务器计算机108以及计算环境100内的其他计算设备(未示出)之间的通信的连接和协议的任何组合。

[0024] 服务器计算机108是托管异常检测器110和数据库112的数字设备。在该实施例中，服务器计算机108可驻留在云架构(例如，公共、混合或私有)中。在该实施例中，异常检测器110驻留在服务器计算机108上。在其他实施例中，异常检测器110可以具有本地地存储在客户端计算机设备102上的程序(未示出)的实例。在其他实施例中，异常检测器110可以是训练多语言神经网络意图分类器的独立程序或系统。在又一些实施例中，异常检测器110可以被存储在任何数目的计算设备上。

[0025] 异常检测器110通过使用神经网络方法考虑请求上下文数据中的依赖性来实现微服务应用的主动异常检测。由异常检测器110提供的解决方案独立于微服务应用(例如，私

有云、公共云或混合云)的部署并支持各种容器编排器(例如,Kubenetes、OpenShift等)。异常检测器110提供用于基于应用和系统行为两者的混合数据收集的机制。在该实施例中,异常检测器110可包括参见图2更详细描述的一个或多个组件。

[0026] 例如,异常检测器110可以接收对包括N个微服务的应用的终端用户请求。在每个微服务实例处,(与异常检测器110相关联的)相应收集代理提取每个相应实例的跟踪数据和规范。异常检测器110的收集器代理随后编译接收到的信息(相应的跟踪数据和规范)并对接收到的信息进行标准化。从那里,收集器代理可将数据推送到队列以供持久化。特征提取模块(在图2中示出和描述)将原始数据转换成请求上下文特征。异常检测器110随后可使用格式化的上下文特征来构建神经网络模型,并随后使用所构建的模型来生成预测。异常检测器110随后可生成主动警报。

[0027] 在该实施例中,异常检测器110可以响应于接收到预测异常行为的请求,从相应的微服务请求附加信息。附加信息可以包括上下文特征,即,表示请求的端到端细节的层级数据结构。上下文特征可以包括一个或多个随意相关的服务和调用路径。上下文特征还可包括每个服务实例处的执行上下文(例如,CPU、加速器、存储器利用率、pod区域、网络流量、I/O请求等)。

[0028] 例如,对附加信息(例如,请求规范)、微服务路径和功能路径的请求。附加信息的示例可以包括与用户相关联的用户名(匿名ID)、公司名称(匿名ID)、延迟(例如,500ms)、区域(例如,欧洲)、浏览器类型、设备类型、操作系统、时间(例如,星期五,2月28日,2020年,下午2:55:02GMT-05:00)。

[0029] 微服务路径的示例可以包括从微服务A到微服务B的路径。例如,与微服务A相关联的集群ID、区域(us)、实例ID、持续时间(100ms)、OS性能(CPU、存储器、盘、网络),以及用于微服务B的相应集群ID、区域(us)、实例ID、持续时间(400ms)、OS性能(CPU、存储器、盘、网络)。

[0030] 调用路径(即,功能路径)的示例可包括一个或多个函数。例如,函数一至函数三:函数一包括持续时间(40ms),资源利用率(20%,100MB,)函数二包括持续时间(60ms),资源利用率(20%,100MB)回到函数一包括持续时间(400ms),资源利用率(20%,100MB)。

[0031] 在该实施例中,异常检测器110提供混合数据收集来请求上下文特征,即,对上下文特征的请求可被发送到不同源或以其他方式从不同源收集。在该实施例中,异常检测器110包括收集代理(在图2中示出和讨论),该收集代理被部署在每个微服务实例内作为侧车(sidecar)(例如,单个Kubernetes Pod的两个容器)并且可从两个不同的源拉取:来自诸如Jaeger和OpenTelemetry之类的微服务的跟踪数据以及微服务运行时的特征(例如,CPU、存储器利用率、网络、其他并置的侧车、Zabbix代理(例如,CPU、磁盘、存储器等)、Istio的Envoy(例如,网络)等)。

[0032] 异常检测器110可以从这些源收集分类数据和数字数据。在该实施例中,分类数据指的是从请求报头或部署主机上的环境变量中提取的请求和微服务实例。在该实施例中,数字数据是指报告花费在每个微服务上的时间及其来自诸如OpenTelemetric或Jaeger的分布式跟踪库的关键函数的数据。以此方式,异常检测器110可以利用该数字数据报告,以适当的权限报告、记录、和检索关于相应系统利用的信息。因而,通过从不同源收集上下文特征,异常检测器110可以实现跨层处理请求的整体视图。

[0033] 异常检测器110随后可使用所收集的上下文特征(即,附加信息)来构建和训练神经网络模型,该神经网络模型可预测在相应微服务应用内处理的未来请求,从而将前述请求上下文特征分层地作为输入来处理。

[0034] 以此方式,(使用所构建的神经网络模型的)异常检测器110可捕捉请求间和请求内因子并使用所捕捉的因子来预测未来请求。在该实施例中,请求间描述请求规范中的特性之间的连接(例如,来自某个区域的用户id的登录请求很可能跟随有来自同一区域用户id的对产品目录页面的获取请求)。在本实施例中,请求内因子考虑单个请求的因子,以理解在处理路径期间哪些服务对于来自随意相关的微服务和功能路径数据的未来请求起着最重要的作用。通过考虑这两个因子,所建立的神经网络模型可捕捉相应微服务和上一步之间的相关性。例如,来自微服务的历史请求可采用两条路径。第一路径可利用具有40ms、15ms和300ms的相应延迟的微服务A、B和C。第二路径可利用分别具有200ms、40ms和1.2s的延迟的微服务A、B和D。构建的神经网络可预测使用微服务A、B和D的路径,在微服务A的延迟高时利用微服务D。例如,微服务A可以具有300ms的延迟,而微服务B可以具有50ms的延迟。在该示例中,异常检测器110可以(使用所构建的神经网络)预测下一个请求应当在具有2s的延迟的微服务D处而不是在具有100ms的延迟的C处理,并且在时间2.35s处,异常检测器110可以发送警报(例如,2.35s＝300ms(A)+50ms(B)+2s(D))。跟踪路径(A-》B-》D))是神经网络模型的预测结果,其捕获A的持续时间与上一次的选择之间的相关性。这是经由神经网络模型的请求(用于预测),该神经网络模型被构建并且稍后关于图3和4示出和描述。具体地,LSTM模型将被训练以学习微服务之间的顺序关系并预测哪一个将是将被使用的下一个。

[0035] 在该实施例中,异常检测器110可以利用控制器(在图2中示出和描述)来解释预测序列并决定是否将发生异常。在该实施例中,控制器对关键性能度量(例如,延迟、吞吐量、失败的RPC调用等)进行加权。在该实施例中,关键性能度量可以由微服务应用的所有者来确定或以其他方式定义。控制器计算统计测量(例如,偏差、百分位),并且确定是否提出主动警报。例如,控制器可以根据以下公式计算偏差:偏差＝|xi-平均值(X)|。在该实施例中,偏差越大,数据集越不稳定,这表示某种异常。在该实施例中,百分位被定义为低于该数目的一定百分比的得分。例如,数字顺序列表的第50百分位数是其中值。

[0036] 在该实施例中,异常检测器110可以响应于所预测的异常行为来生成主动警报。所产生的主动警报可包含为何预测和/或以其他方式标记异常的原因。在该实施例中,主动警报可由异常检测器110的组件(例如,图2中示出和描述的控制器)生成。在该实施例中,控制器可以生成适当的可视化、主动警报、生成根本原因报告、提供资源管理能力和系统模拟。

[0037] 例如,异常检测器110可以生成处理终端用户请求的相应组件的可视化。该请求可被发送到包含以下组件的以下云基础设施:前端服务、路由器服务、调度器服务、适配器服务、场所内基础设施(例如,传统(legacy)代码)、消费者、后端服务、以及包含两个不同位置(例如,美国和欧洲)中的数据库的私有云软件即服务(SaaS)。在该示例中,异常检测器110可以生成请求的每个相应组件和功能路径的可视化,以及生成一个或多个图形图标以在视觉上显示所检测到的根本原因可能是服务(例如,调度器)之一。以这种方式,异常检测器110可以生成异常请求的端到端执行流程的可视化,并将调度器服务器高亮为根本原因。

[0038] 在该实施例中,根本原因报告包含预测的异常服务和可能的原因,以及所生成的

包含推理的主动警报。继续以上示例，根本原因报告可包括对调度器中的异常行为的描述，并产生存在违反服务等级协议的影响终端用户的长延迟的主动警报。

[0039] 在该实施例中，异常检测器110可以提供警告系统管理员并采取适当动作的资源管理能力。例如，如果预测的异常的原因是由于诸如CPU、低存储器、慢网络延迟之类的计算资源不足引起的，则系统管理员可以在其影响应用客户端之前供应更多资源。

[0040] 在该实施例中，异常检测器110还可提供系统模拟。例如，预测结果包含每个微服务处的端到端执行流的细节，包括CPU、存储器、磁盘和网络使用。此类细粒度特征化的跟踪提供对在底层硬件系统上要求的应用的洞察，其可被用作系统模拟器的驱动器以评估潜在云系统设计以学习挑战和权衡(例如，本地对远程、路由流/流量控制、强核对弱核(brawny vs wimpy cores)、延迟要求、卸载益处等)。该过程帮助云系统设计者理解不同可组合硬件组件(诸如来自各种应用的存储、网络、CPU、存储器和加速器)之间的交互。它还有助于分析具有不同硬件配置的潜在益处与降级，并且指导未来云系统的设计决策。

[0041] 在端到端示例中，由异常检测器110处置的系统可接收针对处理的请求。该请求可被发送到包含以下组件的以下云基础设施：前端服务、路由器服务、调度器服务、适配器服务、场所内基础设施(例如，传统代码)、消费者、后端服务、以及包含两个不同位置(例如，美国和欧洲)中的数据库的私有云软件即服务(SaaS)。

[0042] 在第一场景中，请求可由前端服务处理、发送至路由器、至适配器返回至消费者、以及最后至后端组件。在该场景中，异常检测器110可以响应于预测调度器和后端服务经历影响终端用户和违反SLA的长延迟来生成主动警告。通过使用异常检测器110，检测调度器和后端服务中的异常行为，并将其适当地归为引起延迟的服务实例。相反，由于从并发请求收集的混合日志，使用预测模型的当前系统产生不太准确的结果(例如，低准确度)。本发明的各实施例(例如，异常检测器110)与当前方法的不同之处在于请求上下文数据包含将日志分离成个体请求的踪迹。例如，路由器服务正在同时处理十个请求，它们中的四个将被路由到调度器，而其他将被路由到后端。当前方法可仅查看由于并行处理而交错的混合日志数据。因此，当一个或多个请求失败时，难以识别哪一个失败。不同地，异常检测器110提供跟踪数据(即，请求上下文数据)，我们可以标识哪个请求在哪个服务处失败。

[0043] 在利用上述组件的第二场景中，异常检测器110可以预测后端服务正经历来自存储用户信息的数据库的缓慢响应，并且可以生成主动警报，该主动警报告诉用户针对特定用户集合的延迟响应。相反，当前系统难以检测问题以统计聚合度量。在一些场景中，聚合的度量可能误导监控组件。例如，低于某个阈值的平均延迟不一定意味着系统是健康的。在这个实例中，如果90％的流量被路由到欧洲(EU)DB并且10％被路由到美国(US)DB。当EU DB正常并且US DB服务异常时，平均延迟将仍然看起来正常，因为90％的请求具有正常的延迟。不同地，我们的模型(例如，异常检测器110)考虑个体踪迹的延迟，从而我们可以识别到US DB的执行路径上的异常。

[0044] 在利用上述组件的第三场景中，异常检测器110可以预测调度器服务发起的作业由于传统代码处的性能降级而不能完成，并且生成后端延迟接收来自消费者的结果的警告。相反，当前系统难以使用生产者和消费者的日志的度量对异步关系建模。当前的系统使用日志数据来训练机器学习模型。如前所述，从个体收集的日志数据被交织，使得因果关系难以导出。相反，由于请求上下文建立在踪迹之上，所以异常检测器110避免这个问题。

[0045] 异常检测器110还可利用预测的结果来执行根本原因分析、资源管理和系统模拟。例如，预测的结果可以用于驱动系统模拟器来理解来自各种硬件配置的潜在益处和降级，以及指导针对未来云系统的设计决策。

[0046] 数据库112存储所接收的信息并且可以表示向异常检测器110或公共可用数据库提供经许可的访问的一个或多个数据库。通常，可以使用本领域已知的任何非易失性存储介质来实现数据库112。例如，数据库112可以用磁带库、光学库、一个或多个独立硬盘驱动器、或独立磁盘冗余阵列(RAID)中的多个硬盘驱动器来实现。在该实施例中，数据库112存储在服务器计算机108上。

[0047] 图2描绘了根据本发明的实施例的用于微服务的异常检测器的示例框图200。

[0048] 该示例图示出异常检测器110的一个或多个部件。在一些实施例中，异常检测器110可以包括具有相应的微服务和收集代理的一个或多个主机，然而，应当理解，异常检测器110可以跨云架构访问微服务和收集代理。

[0049] 在该示例中，异常检测器可包括主机202A、主机202B至202N。每个主机可以具有相应的微服务和收集代理，(例如，相应的微服务204A-N和收集代理206A-N)

[0050] 在该示例中，异常检测器110可以经由收集代理206A来接收终端用户请求微服务204A。在该示例中，收集代理206可以接收来自终端用户的请求，并且还接收来自一个或多个其他组件(例如，其他并置的侧车、Zabbix代理(例如，CPU、磁盘、存储器等)、Istio的Envoy(例如，网络)等)的请求。

[0051] 收集代理206A负责收集请求并提取每个相应实例的跟踪数据和规范。在该实施例中，相应的收集代理可以与异常检测器110的收集器模块(例如，收集器模块206)对接。收集器模块206负责编译接收到的信息(各自的跟踪数据和规范)。收集器模块206然后可以使用标准化模块210来标准化数据，即，标准化模块210将数据标准化成一致的格式(例如，JSON或公共数据结构)。收集器模块206然后可以将经编译的信息推送到队列中以供持久化。

[0052] 特征提取模块213可接着访问队列中的数据，且从经编译的数据提取上下文特征。换言之，特征提取模块210将原始数据转换成请求上下文特征。例如，请求上下文特征(即，请求规范)可以包括：用户名(匿名ID)、公司名(匿名ID)、延迟(500ms)、区域(EU)、浏览器(Firefox)、设备(iOS)、时间(星期五，2020年2月28日，2:55:02PM-GMT-05:00)、相应的微服务路径(例如，从微服务A到微服务B的路径)。例如，与微服务A相关联的集群ID、区域(us)、实例ID、持续时间(100ms)、OS性能(CPU、存储器、盘、网络)和微服务B的相应集群ID、区域(us)、实例ID、持续时间(400ms)、OS性能(CPU、存储器、盘、网络)，以及功能路径(例如，功能一至功能三：功能一包括持续时间(40ms)，资源利用率(20％，100MB，)功能二包括持续时间(60ms)，资源利用率(20％，100MB)返回到功能一，包括持续时间(400ms)，资源利用率(20％，100MB))。

[0053] 异常检测器110随后可使用格式化的上下文特征来使用神经网络模块214(在图3和4中示出和描述)来构建神经网络模型。控制器模块216随后可使用所构建的神经网络模型来生成预测，并且可生成适当的可视化、主动警报、生成根本原因报告、提供资源管理能力和系统模拟。

[0054] 图3描绘了根据本发明的实施例的神经网络模型的设计的示例框图300。

[0055] 具体地，框图300描绘了神经网络的设计(省略一些隐藏层)。输入是一系列请求的

请求规范。请求内嵌入层的输入S1是图4中示出和描述的微服务路径神经网络模型的输出。

[0056] 在该示例中，异常检测器110接收输入302A、302B至302N(r1规范)。例如，请求输入(即，附加信息)可包括在指定时间(例如，时间窗口T)期间收集的上下文分层结构跟踪数据。该请求输入可以包括请求规范、微服务路径和功能路径。请求规范的附加信息的示例可以包括与用户相关联的用户名(匿名ID)、公司名称(匿名ID)、延迟(例如，500ms)、区域(例如，欧洲)、浏览器类型、设备类型、操作系统、时间(例如，星期五，2月28日，2020年，下午2:55:02GMT-05:00)。

[0057] 微服务路径的示例可以包括从微服务A到微服务B的路径。例如，与微服务A相关联的集群ID、区域(us)、实例ID、持续时间(100ms)、OS性能(CPU、存储器、盘、网络)，以及用于微服务B的相应集群ID、区域(us)、实例ID、持续时间(400ms)、OS性能(CPU、存储器、盘、网络)。

[0058] 调用路径(即，功能路径)的示例可包括一个或多个函数。例如，函数一至函数三：函数一包括持续时间(40ms)，资源利用率(20％，100MB，)函数二包括持续时间(60ms)，资源利用率(20％，100MB)回到函数一包括持续时间(400ms)，资源利用率(20％，100MB)。

[0059] 所接收的输入随后在框320中被处理以用于请求规范嵌入(例如，r1和a1，分别为304a-n和306a-n)。在该实施例中，“r1”是请求规范中的字符串部分(例如，用户名、浏览器类型等)的嵌入结果，而“a1”是指与请求规范相关联的数字部分。在该实施例中，异常检测器110将嵌入的结果与请求规范的数字部分(例如，延迟，称为a1-an)连接。

[0060] 异常检测器随后可将嵌入式请求规范与分别被称为308a-n和310a-n的组件b1和s1组合。在本实施例中，b1-bN是嵌入请求规范的输出。在该实施例中，S1是图4中描述的模型的输出。在该实施例中，S1表示单个请求的端到端执行流的建模输出。

[0061] 该过程在框330中继续请求内嵌入。请求内因子包括B1 S1和C1。在该实施例中，B1、S1和C1与单个请求规范相关。类似地，B2、S2和C1与另一请求规范相关。C1是将B1和S1的组合转换为向量的嵌入层(称为312A-N)。

[0062] 该过程继续添加包括块340和块350的请求间因子(例如，LTSM 340和全连接(Dense)350)。在框340中，上下文特征通过在深度学习领域使用的长短期(LSTM)架构馈送，并增加D1，分别称为314A-N。在本实施例中，D1是LSTM模型的单个单元。回想起C1、C2、…CN是单独请求的建模输出。异常检测器110使用LSTM模型来学习请求之间的请求间关系。在本实施例中，D1-Dn是LSTM模型的单元。最后，在全连接350中，添加E1，称为316A-N。在这个实施方式中，E1-EN是全连接的网络的单元，其减少输入的尺寸以便发现其内部相关性。所得输出为Y_1、Y_2至Y_N，分别标记为318_{A-N}。

[0063] 图4描绘了根据本发明的实施例的捕获针对个体请求的请求内因子的神经网络模型的示例框图400。

[0064] 输入(例如，分别被称为402A、402B、402C和402N的$F_{1,1}$、$F_{1,2}$、$F_{2,1}$和F_{B1})是一系列请求的请求规范中的功能的描述。异常检测器110取得接收到的输入并执行请求规范嵌入(例如，框420)。在该实施例中，$G_{1,1}$、$G_{1,2}$、$G_{2,1}$和$G_{B,1}$被称为404A、404B、404C至404N，而$H_{1,1}$、$H_{1,2}$、$H_{2,1}$和$H_{B,1}$被分别称为406A、406B、406C和406N。$G_{1,1}$、$G_{1,2}$是函数$F_{1,1}$中的字符串部分的嵌入层。类似地，$G_{2,1}$是函数$F_{2,1}$中的串部分的嵌入单元。$H_{1,1}$表示$G_{1,1}$与$F_{1,1}$的数字部分的串联。总体而言，404A-N和406A-N以与图3中描述的304A-N和306A-N类似的方式起作用。

[0065] 在该实施例中，在框430中，通过长短期存储器(LSTM)、人工递归神经网络(RNN)来馈送嵌入的请求规范，并添加相应的$K_{1,1}$、$K_{1,2}$、$K_{2,1}$和$K_{B,1}$(即，LTSM模型的单元分别被称为408A、408B、408C和408N)。

[0066] 处理继续到块440以进行微服务嵌入，其中，分别添加M_1、M_2和M_B以及O_1、O_2和O_B。M_1、M_2和M_B被称为块410A、410B和410N，块410A、410B和410N是表示B微服务的LTSM模型的输出(例如，块430)，而O_1、O_2和O_B被分别称为块412A、412B和412N，并且引用B微服务的规范的嵌入。

[0067] 然后，处理继续至块450，其中，块440的结果通过分别添加P_1、P_2和P_B的另一LTSM层馈送。P_1、P_2和P_B分别被标记为框414A、414B和414N。在本实施方式中，P_1、P_2和P_B是块450的LTSM模型的单元。

[0068] 框450的所得输出馈送通过框460。框460是全连接层，该全连接层提供来自前一层的特征的所有组合的学习特征并且加上Q_1、Q_2和Q_B，分别被标记为416A、416B和416N。

[0069] 在该实施例中，Z_1、Z_2和Z_N(分别标记为418_A、418_B和418_N)是框图400的工作流的结果输出。总的来说，418_A、418_B和418_N表示单个请求的端到端执行流程的建模输出。418_B和418_N被标记为S1，并且被描绘为并入图3所描述的模型中。

[0070] 图5是描绘根据本发明实施例的用于训练端到端语音、多语言意图分类器的操作步骤的流程图500。

[0071] 在步骤502中，异常检测器110接收信息。在该实施例中，所接收的信息可包括对包括N个微服务的应用的终端用户请求。例如，终端用户请求是通过用户对前端服务的需求触发的请求。例如，当用户访问网页并点击登录按钮时，生成对应用的登录请求。

[0072] 在该实施例中，异常检测器110接收来自客户端计算设备102的请求。在其他实施例中，异常检测器110可以从计算环境100的一个或多个其他组件接收信息。

[0073] 在步骤504中，异常检测器110从接收的信息生成上下文信息。在该实施例中，异常检测器110通过请求附加信息和创建表示接收到的请求的端到端细节的层级数据结构来从接收到的请求生成上下文信息。

[0074] 具体地，异常检测器110可以请求附加信息(例如，请求规范)，可以包括与用户相关联的用户名(匿名ID)、公司名称(匿名ID)、延迟(例如，500ms)、区域(例如，欧洲)、浏览器类型、设备类型、操作系统、时间(例如，星期五，2月28日，2020年下午，2:55:02PMGMT-05:00)、微服务路径和功能路径。

[0075] 对上下文特征的请求可以被发送到不同的源或者以其他方式从不同的源收集。在该实施例中，异常检测器110包括收集代理(在图2中示出和讨论)，该收集代理被部署在每个微服务实例内作为侧车(例如，单个Kubernetes Pod的两个容器)并且可从两个不同的源拉取：来自诸如Jaeger和OpenTelemetry之类的微服务的跟踪数据以及微服务运行时的特征(例如，CPU、存储器利用率、网络、其他并置的侧车、Zabbix代理(例如，CPU、磁盘、存储器等)、Istio的Envoy(例如，网络)等)。

[0076] 异常检测器110可以从这些源收集分类数据和数字数据。在该实施例中，分类数据指的是从请求报头或部署主机上的环境变量中提取的请求和微服务实例。在该实施例中，数字数据是指报告花费在每个微服务上的时间及其来自诸如OpenTelemetric或Jaeger的分布式跟踪库的关键函数的数据。以此方式，异常检测器110可以利用该数字数据报告，以

适当的权限报告、记录、和检索关于相应系统利用的信息。因而，通过从不同源收集上下文特征，异常检测器110可以实现跨层处理请求的整体视图。

[0077] 在步骤506中，异常检测器110基于所生成的上下文信息来训练神经网络。在该实施例中，异常检测器110基于所生成的包括请求间和请求内因子的上下文信息来训练神经网络。如前所述，请求间描述请求规范中的特性之间的连接(例如，来自某个区域的用户id的登录请求很可能跟随有来自同一区域用户id的对产品目录页面的获取请求)。相反，请求内因子考虑个别请求的因子，以理解在处理路径期间哪些服务对于来自随意相关的微服务和功能路径数据的未来请求起着最重要的作用。通过考虑这两个因子，所建立的神经网络模型可捕捉相应微服务和上一步之间的相关性。以此方式，经训练的神经网络可预测下一系列请求和它们的上下文请求看起来像什么。然后，基于该预测，控制器模块将确定是否将存在任何异常。

[0078] 在步骤508中，异常检测器110使用经训练的神经网络模型来预测异常行为。例如，异常检测器110可以预测诸如SLA违反(例如，在接下来的十分钟内，尾部延迟将增加)、将受影响的用户(例如，U、南部区域中的用户的子集)和请求子集的影响(例如，检索分析结果将失败)之类的异常。

[0079] 在步骤510中，异常检测器110基于所预测的异常行为来采取适当的动作。在该实施例中，适当的动作可通过生成主动警报、生成根本原因报告、提供资源管理能力和系统模拟来实现。例如，异常检测器110随后可基于该预测来确定是否发送主动警报。在该实施例中，异常检测器110可以响应于预测异常而自动生成主动警报。在另一实施例中，异常检测器可以为所预测的异常生成加权分数，并且响应于所预测的异常满足或超过异常行为的阈值，生成主动警报。

[0080] 例如，主动警报可包含以下预测：SLA违反(例如，在接下来的十分钟内，尾部延迟将增加)、将受影响的用户(例如，U、南区中的用户子集)以及请求子集的影响(例如，检索分析结果将失败)。

[0081] 根本原因报告的示例可包括失败的微服务实例的标识以及失败的原因。例如，缓慢的数据库连接、不足的计算资源等。

[0082] 在一些实施例中，资源管理可包括推荐的修复。例如，异常检测器110可以推荐在具有较高容量的节点处提供微服务实例，增加后端与数据库之间的网络带宽，添加具有更高功率CPU的节点等。

[0083] 图6示出了根据本发明的实施例的示例图600。

[0084] 例如，图6示出了具有编码器和解码器部分的序列到序列(seq2seq)模型、它们的输入和输出(表示上述方法)的概述。编码器(例如，块602)和解码器(例如，块604)部分两者都是基于RNN的并且能够消耗和返回对应于多个时间步长的输出序列。模型从先前N个值获得输入，并且它返回接下来的N个预测。N是超参数并且在该图中经验地设置为10分钟。在该图的中间是分层的基于RNN的异常检测器神经网络，包括三个主要组件：请求内因子、请求间因子、和嵌入。

[0085] 具体地，图6中的图是编码器-解码器架构(如被称为seq2Seq模型)。在该实施例中，$X, X_1, X_2, \cdots, X_n$表示对模型的输入，是一系列请求的请求上下文数据。在该实施例中，$Y, Y_1, Y_2, \cdots, Y_n$是模型的输出，是模型的预测。该模型的内部架构在图3和图4中已进行了详细

讨论。

[0086] 图7A和7B示出了根据本发明的实施例的示例数据收集代码。

[0087] 具体地，图7A描绘了示例数据收集代码700，其是相应微服务中的示例应用代码。

[0088] 关于图7B，7B描绘了示例数据收集代码750。具体地，示例数据收集代码750表示收集代理中的代码。

[0089] 图8描绘了根据本发明的实施例的图1的计算环境100内的计算系统的组件的框图。应当理解，图8仅提供一个实现方式的图示并且不暗示关于其中可以实现不同实施例的环境的任何限制。可以对所描绘的环境做出许多修改。

[0090] 在此描述的程序是基于应用在本发明的具体实施例中实施的来识别的。然而，应当理解，本文中的任何特定程序术语仅为了方便而使用，并且因此本发明不应局限于仅在由这样的术语标识和/或暗示的任何特定应用中使用。

[0091] 计算机系统800包括通信构造802，其提供高速缓存816、存储器806、永久性存储器808、通信单元812和输入/输出(I/O)接口814之间的通信。通信构造802可用被设计用于在处理器(诸如微处理器、通信和网络处理器等)、系统存储器、外围设备和系统内的任何其他硬件组件之间传递数据和/或控制信息的任何架构来实现。例如，通信构造802可用一个或多个总线或纵横开关来实现。

[0092] 存储器806和永久性存储器808是计算机可读存储介质。在该实施例中，存储器806包括随机存取存储器(RAM)。一般而言，存储器806可包括任何合适的易失性或非易失性计算机可读存储介质。高速缓存816是快速存储器，其通过保存来自存储器806的最近访问的数据和接近访问的数据的数据来增强(多个)计算机处理器804的性能。

[0093] 异常检测器110(未示出)可被存储在持久存储808和存储器806中以供相应的计算机处理器804中的一个或多个经由高速缓存816执行。在实施例中，永久性存储器808包括磁性硬盘驱动器。可替代地，或除了磁性硬盘驱动之外，永久性存储装置808可以包括固态硬盘驱动、半导体存储装置、只读存储器(ROM)、可擦除可编程只读存储器(EPROM)、闪存、或能够存储程序指令或数字信息的任何其他计算机可读存储介质。

[0094] 永久性存储器808使用的介质也可以是可移动的。例如，可移动硬盘驱动器可以用于永久性储存器808。其他示例包括光盘和磁盘、拇指驱动器和智能卡，它们被插入到驱动器中以便转移到也是永久存储装置808的一部分的另一计算机可读存储媒质上。

[0095] 在这些示例中，通信单元812提供与其他数据处理系统或设备的通信。在这些示例中，通信单元812包括一个或多个网络接口卡。通信单元812可以通过使用物理和/或无线通信链路提供通信。异常检测器110可以通过通信单元812下载到永久性存储器808。

[0096] (多个)I/O接口814允许与可连接到客户端计算设备和/或服务器计算机的其他设备进行数据的输入和输出。例如，I/O接口814可提供到外部设备820(诸如键盘、小键盘、触摸屏和/或一些其他合适的输入设备)的连接。外部设备820还可包括便携式计算机可读存储介质，诸如例如拇指驱动器、便携式光盘或磁盘、以及存储卡。用于实施本发明的实施例的软件和数据(例如，异常检测器110)可存储在这种便携式计算机可读存储介质上并且可经由I/O接口814加载到永久性存储装置808上。I/O接口814还连接到显示器822。

[0097] 显示器822提供向用户显示数据的机制，并且可以是例如计算机监视器。

[0098] 本发明可以是系统、方法和/或计算机程序产品。计算机程序产品可包括其上具有

用于使处理器执行本发明的各方面的计算机可读程序指令的计算机可读存储媒质(或多个媒质)。

[0099] 计算机可读存储介质可以是能够保留和存储由指令执行设备使用的指令的任何有形设备。计算机可读存储媒质可以是,例如但不限于,电子存储设备、磁存储设备、光存储设备、电磁存储设备、半导体存储设备、或者上述的任意合适的组合。计算机可读存储媒质的更具体示例的非穷尽列表包括以下各项:便携式计算机盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦式可编程只读存储器(EPROM或闪存)、静态随机存取存储器(SRAM)、便携式紧凑盘只读存储器(CD-ROM)、数字通用盘(DVD)、记忆棒、软盘、诸如穿孔卡之类的机械编码设备或具有记录在其上的指令的槽中的凸出结构、以及上述各项的任何合适的组合。如本文所使用的计算机可读存储媒体不应被解释为暂时性信号本身,例如无线电波或其他自由传播的电磁波、通过波导或其他传输媒体传播的电磁波(例如,穿过光纤电缆的光脉冲)或通过电线发射的电信号。

[0100] 本文中所描述的计算机可读程序指令可以经由网络(例如,互联网、局域网、广域网和/或无线网络)从计算机可读存储介质下载到相应的计算/处理设备,或者下载到外部计算机或外部存储设备。网络可以包括铜传输电缆、光传输纤维、无线传输、路由器、防火墙、交换机、网关计算机和/或边缘服务器。每个计算/处理设备中的网络适配器卡或网络接口接收来自网络的计算机可读程序指令,并转发计算机可读程序指令以存储在相应计算/处理设备内的计算机可读存储媒质中。

[0101] 用于执行本发明的操作的计算机可读程序指令可以是汇编指令、指令集架构(ISA)指令、机器指令、机器相关指令、微代码、固件指令、状态设置数据、或以一种或多种程序设计语言的任何组合编写的源代码或目标代码,这些程序设计语言包括面向对象的程序设计语言(诸如Smalltalk、C++等)和常规的过程式程序设计语言(诸如“C”程序设计语言或类似程序设计语言)。计算机可读程序指令可以完全地在用户计算机上执行、部分在用户计算机上执行、作为独立软件包执行、部分在用户计算机上部分在远程计算机上执行或者完全在远程计算机或服务器上执行。在后一种情况下,远程计算机可通过任何类型的网络(包括局域网(LAN)或广域网(WAN))连接到用户计算机,或者可连接到外部计算机(例如,使用互联网服务提供商通过互联网)。在一些实施例中,包括例如可编程逻辑电路、现场可编程门阵列(FPGA)或可编程逻辑阵列(PLA)的电子电路可以通过利用计算机可读程序指令的状态信息来使电子电路个性化来执行计算机可读程序指令,以便执行本发明的各方面。

[0102] 下面将参照根据本发明实施例的方法、装置(系统)和计算机程序产品的流程图和/或框图描述本发明。应当理解,流程图和/或框图的每个方框以及流程图和/或框图中各方框的组合,都可以由计算机可读程序指令实现。

[0103] 这些计算机可读程序指令可被提供给通用计算机、专用计算机或其他可编程数据处理装置的处理器以产生机器,使得经由计算机或其他可编程数据处理装置的处理器执行的指令创建用于实现在流程图和/或框图的或多个框中指定的功能/动作的装置。也可以把这些计算机可读程序指令存储在计算机可读存储媒质中,这些指令使得计算机、可编程数据处理装置、和/或其他设备以特定方式工作,从而,其中存储有指令的计算机可读存储媒质包括包含实现流程图和/或框图中的或多个方框中规定的功能/动作的方面的指令的制造品。

[0104] 也可以把计算机可读程序指令加载到计算机、其他可编程数据处理装置、或其他设备上，使得在计算机、其他可编程装置或其他设备上执行一系列操作步骤，以产生计算机实现的处理，使得在计算机、其他可编程装置或其他设备上执行的指令实现流程图和/或框图中的或多个方框中规定的功能/动作。

[0105] 附图中的流程图和框图示出了根据本发明的不同实施例的系统、方法和计算机程序产品的可能实现方式的架构、功能和操作。对此，流程图或框图中的每个方框可表示包括用于实现指定的(多个)逻辑功能的一个或多个可执行指令的模块、片段或指令的一部分。在一些备选实现中，框中标注的功能可以不按照图中标注的顺序发生。例如，取决于所涉及的功能，连续示出的两个块实际上可以基本上同时执行，或者这些块有时可以以相反的顺序执行。也要注意的是，框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合，可以用执行规定的功能或动作或执行专用硬件与计算机指令的组合的专用的基于硬件的系统来实现。

[0106] 已经出于说明的目的呈现了本发明的各种实施方式的描述，但并不旨在是详尽的或者限于所公开的实施方式。在不背离本发明的范围的情况下，许多修改和变化对于本领域普通技术人员来说是显而易见的。在此所使用的术语被选择来最好地解释实施例的原理、实际应用、或优于市场中所发现的技术的技术改进、或使得本领域普通技术人员能够理解在此所披露的实施例。

[0107] 进一步的评论和/或实施例

[0108] 本发明的一些实施例认识到以下事实、潜在问题和/或潜在领域，用于相对于现有技术的改进：微服务架构对于在混合云环境中部署的应用是吸引人的，因为松散耦合的组件提供更好的可扩展性、灵活性、加速的开发者生产力等。为了避免由SLA违背引起的严重的财务和业务损失，管理微服务应用的最关键的任务之一是在某些时间步骤中有效且高效地检测和诊断异常，使得DevOps/SRE可以采取进一步的行动以便及时地解决底层问题。然而，用于对检测到的异常发出主动警报的现有方法对于微服务应用仍然不是有效的，因为它们不考虑在来自解耦服务和终端用户的请求的多变量时间序列数据中隐藏的空间和时间依赖性。

[0109] 本发明的一些实施例可以包括以下特征、特性和/或优点中的一个或多个：在模型中学习尾部延迟问题并且帮助在发生潜在异常之前预测潜在异常。

[0110] 本发明的各实施例预测异常并标识微服务应用的根本情况。在现有的异常预测工作中，本发明的实施例是首先进行双任务来预测请求模式及其路径(即，请求所经历的服务)。本发明的实施例设计收集代理以从应用部署收集数据。该系统支持微服务应用在不同环境(私有、公共和混合)中部署。

[0111] 本发明的各实施例定义请求上下文特征的概念，数据结构包括请求的三级信息：请求规范、微服务路径、功能路径。该提出的特征集成了请求间因子和请求内因子、影响传入请求的性能和处理路径的两个历史数据。

[0112] 本发明的各实施例设计分层神经网络模型以集成请求上下文特征的训练数据。该模型是基于具有异构数据和关注机制的嵌入的seq2Seq架构，这导致结果的一定水平的可解释性。

[0113] 专用系统跟踪信息的独特益处是双重的。我们利用带时间戳的系统利用信息来理

解和预测系统资源要求,以进一步引导系统管理员重新分配资源以满足QoS要求。我们还使用从y上的应用导出的详细细粒度系统表征,通过系统模拟来理解不同硬件含意和权衡,并将此类经验用作用于未来云系统设计的输入。

[0114] 本发明的实施例通过利用深度学习水平地和垂直地分析在请求文本数据中可用的上述依赖性而实现微服务应用的主动警报和异常诊断。所提出的方法解决了两个具体问题:(1)将存在在从当前时刻流逝的特定时间步长处发生的任何性能异常(例如,SLA违反、增加的尾部延迟)吗?以及(2)如果(1)是真的,导致异常的最可能的微服务是什么?第一个问题是关于异常预测,第二个问题告诉预测的异常的根本原因。

[0115] 解决方案

[0116] 主动警报和异常诊断的问题可被视为关于一组微服务如何协作地处理未来请求的预测任务。我们提出的技术是一种神经网络方法,用于整合历史请求的详细特性,包括其规范和沿路径的每个微服务实例的跟踪信息。神经网络模型可以预测任何异常(例如,尾部延迟、SLA违反)是否将发生以及什么将是根本原因。该解决方案独立于微服务应用(私有云、公共云或混合)的部署并支持各种容器编排器,例如,Kubernetes、OpenShift。

[0117] 关键思想

[0118] 关键思想1:我们介绍请求上下文特征的概念,表示请求的端到端细节的层级数据结构,包括因果相关的服务和调用路径,以及每个微服务处的执行上下文(例如,CPU、加速器、存储器利用率、pod区域、网络浏览、IO请求等)。请求上下文特征由三类信息组成:请求规范、微服务路径和功能路径(6.2节中的细节)。每一类别包含具有异构形式(例如标量、向量、类别)的数据。那些收集的特征点将作为训练数据被提供给神经网络。

[0119] 关键构思2:我们开发了一种用于从不同源收集请求上下文特征的数据的方法(6.1节)。从请求报头或部署主机上的环境变量中提取描述请求和微服务实例的类别数据。报告花费在每个微服务上的时间及其关键功能的数字数据来自诸如OpenTelemetry或Jaeger的分布式跟踪库,而通过利用适当权限检索关于系统利用的信息来记录报告资源使用的数据。结果,请求上下文特征提供了跨层处理请求的整体视图。

[0120] 关键构思3:我们构建神经网络模型以通过将上面提到的请求上下文特征处理为分级输入来预测在微服务应用内如何处理未来的请求。我们相信请求处理预测是依赖于长距离的顺序问题。即,在不久的将来处理请求依赖于两组因子:请求间因子和请求内因子。请求间因子描述请求规范中的特性之间的连接,诸如http方法、用户名、区域。例如,来自某个区域的用户ID的登录请求很可能跟随有来自相同区域和用户ID的产品目录页面的获取请求。请求内因子考虑个别请求的因子。在处理请求时,应用的微服务通过在彼此之间发送RPC调用来协作。进一步,由于每个微服务通常具有许多副本,因此并非所有实例都出现在调用路径中。有效模型应当能够理解在处理路径期间哪些服务对于来自任意相关的微服务和功能路径数据的未来请求起着最重要的作用。在训练过程期间,所有上述因子由所提出的模型捕获。

[0121] 关键思想4:在监测期间,模型一次以一次的步骤生成预测请求的表示,捕捉复杂的请求间和请求内依赖性。创建控制器来解释预测序列:查看关键性能度量(例如,延迟),计算统计测量(例如,偏差、百分位),以及确定是否提出警告。一旦控制器决定提出,根本原因分析模块则解释由当前趋势补充的顺序表示,以精确指出根本原因(例如,区域中特定微

服务实例上的存储器不足、特定微服务实例与后端存储之间的缓慢连接)。

[0122] 启发性实施例

[0123] 我们描述了作为由4个服务组成的微服务应用的预测问题的启发性示例。每个请求必须由A和B处理,然后由C或D处理。在该特定场景中,存在两个历史请求;业务路径为A→B→C和A→B→D。如果我们仅考虑这些请求的序列(即,请求间因子)来预测下一请求及其路径,则结果是A→B→C。从请求间因子学习的模型将请求序列认为是预测过程中的重要特征。假定C和D由于负载平衡的某种影响而交替地出现在历史数据中,结果是合理的并且预测的总延迟＜1s。另一方面,我们所建议的模型智能地保留了沿服务路径的延迟的更多关注,这可能是由于服务实例A处的处理时间增加以及A和最后一跳的选择之间的相关性。因此,它可以成功地预测正确的下一请求和其路径A→B→D,因为如果A处的延迟高,服务D更可能被选择。因为所预测的请求的总延迟是2.3s,其大于阈值,比如1.5s,所以主动警报将被发送到SRE。为了做出正确的预测,我们需要联合地考虑各个请求中的请求间因子和请求内因子,这些因子可以从请求路径的详细信息(诸如跟踪数据、资源利用率以及规范)发现。

[0124] 描述

[0125] 本节介绍了我们提出的解决微型服务应用的主动报警和异常诊断问题的方法和技术细节。在第一阶段中,我们收集针对正常和异常行为两者的一系列请求的跟踪数据和规范,并且准备它们用于特征提取。在该第二阶段中,我们从所收集的数据组装请求上下文特征并且生成神经网络模型。第三阶段负责使用先前训练的模型来预测异常,并且呈现根本原因的列表。

[0126] 如先前在图2中所讨论的,所提出的系统的高级架构,其中,应用由N个微服务组成,具有我们定制设计的收集代理以及模型创建和预测流水线。该部分的剩余部分详细解释端到端ow。

[0127] 数据收集

[0128] 首先(如在流程图500、步骤502-504中所描述的),收集代理从共址的微服务中收集跟踪数据。这对微服务和收集代理在单个Kubernetes pod的分离的容器中运行。微服务运行应用代码以处理请求并且将它们传递到下游服务。此外,收集代理可以聚合来自诸如Zabbix代理或Istio的Envoy代理的侧车的重要系统信息。

[0129] 在微服务内部运行的应用代码使用一些分布式跟踪库(如Jaeger或开Open Telemetry)来记录花费在对业务逻辑至关重要的功能上的时间,并以UDP分组将跟踪数据发送到收集代理。注意,所提出的方法要求在前端服务处捕获用户请求的规范仅一次(例如,参见先前讨论的图7A)。除了微服务中的跟踪信息之外,收集代理不仅必须获取微服务实例的静态配置,而且必须获取在从微服务接收跟踪时的动态资源利用(参见例如前面讨论的图7B)。如上所述,可以从侧车检索这种数据。收集代理将这些数据成批地放置并将它们递送到集中式收集器。

[0130] 收集器被实现为无状态服务器,因此其可被缩放到许多副本。收集器接收跟踪数据和请求的规范,将它们标准化为某一共同表示,并且推送到队列。队列的示例是Kafka,是为处理实时数据馈送(每秒高达百万次写入)提供高吞吐量、低延迟平台的开源软件。

[0131] 然后,异常检测器可以从队列拉到特征提取模块,该特征提取模块被开发成Flink框架顶上的基于流的作业。特征提取的作业是将所收集的数据转换成请求上下文特征的形

式。

[0132] 特征描述

[0133] 我们将收集的特征总结为三类:请求规范、微服务路径、功能路径。请求规范是静态的并且包括请求的自描述信息,最重要的是其在构成应用的微服务集合上的端到端延迟。收集微服务路径特征和功能路径特征作为因果相关的数据,以描述请求的处理路径。图6示出了在时间窗口期间在每个步长收集的层级数据。

[0134] 神经网络模型

[0135] 我们的神经网络模型的设计以seq2Seq架构为基础。如之前在图6中所描述的,神经网络模型包括编码器和解码器部分、它们的输入和输出。编码器和解码器部分都是基于RNN的并且能够消耗和返回对应于多个时间步长的输出序列。模型从先前N个值获得输入,并且它返回接下来的N个预测。N是超参数并且根据经验设定为10分钟。在该图的中间是分层的基于RNN的异常检测器神经网络,该异常检测器神经网络包括三个主要组件:请求内因子、请求间因子、和嵌入。该部分的其余部分描述了神经网络的细节。

[0136] 如之前提到的,图3展示了神经网络的设计。对于请求内因子,我们将序列微服务路径特征和相应的请求规范进行组合。在图4中详述了微服务路径特征,图4是另一基于RNN的网络。对于请求间因子,我们将请求序列的请求内因子馈送到用于训练请求间模式的另一RNN层(例如,LSTM)。在整个网络中,我们应用不同的嵌入层(例如,word2vec,ELMO)来将异构数据转换为N维向量(例如,N=300)。分层请求预测神经网络具有学习请求间和请求内模式对未来请求的处理的影响的能力。如之前所强调的,本发明的实施例旨在预测未来请求的规范以及它们通过应用的微服务实例的路径。

[0137] 监测与启发

[0138] 我们的主动异常检测问题包含两个主要任务:用它们的详细服务路径预测未来请求以及基于该预测来预测SLA违反(图5中的步骤508)。第一个由预测模块制定(例如,图5的步骤510)。在监视阶段期间,系统连续地从运行的应用收集请求上下文数据并且将它们摄取到预测模块。这些数据被馈送至从存储中取得的神经网络模型中。预测模块的输出是具有它们的预测的执行细节的请求序列,其将在下一W_t秒中发生。例如,我们由于经验试验而将W_t设置为500ms,使得自动资源划分软件有机会采取行动。

[0139] 对于确定主动警报的第二任务,我们整合控制器以解释来自预测模块的输出。如图2以及图5中的步骤510所示,控制器具有多个功能。关于主动警报,我们计算预测延迟的尾部。如果结果大于某个阈值,则将提出主动警报。预测结果的细节将进一步用于复杂的任务,如根本原因分析、资源管理、系统模拟。

[0140] 系统模拟:图3的输出包含来自Zabbix代理的实时应用的详细系统(包括CPU、存储器、磁盘和网络使用)跟踪信息。如图1中所讨论的,系统模拟,此类细粒度的特征化跟踪提供对在底层硬件系统上要求的应用的洞察,这可进一步被用作系统模拟器的驱动程序来评估潜在的云系统设计以学习挑战和权衡。该过程帮助云系统设计者理解不同可组合硬件组件(诸如来自各种应用的存储、网络、CPU、存储器和加速器)之间的交互。它还有助于分析具有不同硬件配置的潜在益处与降级,并且指导未来云系统的设计决策。

[0141] 定义

[0142] 本发明:不应被视为由术语“本发明”描述的主题由提交的权利要求或由在专利审

查之后最终发布的权利要求覆盖的绝对指示;虽然术语“本发明”用于帮助读者获得普遍的感觉,在此对这些感觉的披露内容被认为是潜在新的,如由术语“本发明”的使用所指示的这种理解是暂时性的和临时性的,并且在专利审查过程中随着相关信息被开发并且随着权利要求被潜在修改而经历改变。

[0143] 实施例:参见以上“本发明”的定义-相似的注意事项适用于术语“实施例”。

[0144] 和/或:包括性的或;例如,A、B“和/或”C表示A或B或C中的至少一个是真实的并且适用的。

[0145] 包括/包含/含有:除非另有明确说明,是指“包括但不限于”。

[0146] 用户/订户:包括但不必限于以下各项:(i)单个个体人;(ii)具有足够智能以充当用户或订户的人工智能实体;和/或(iii)一组相关用户或订户。

[0147] 模块/子模块:可操作地工作以完成某种功能的任何硬件、固件和/或软件组,而不考虑模块是:(i)在单个本地附近;(ii)分布在广域上;(iii)位于较大软件代码片段内的单个邻近区域中;(iv)位于单个软件代码片段内;(v)位于单个存储设备、存储器或介质中;(vi)机械连接;(vii)电连接;和/或(viii)数据通信连接。

[0148] 计算机:具有显著数据处理和/或机器可读指令读取能力的任何装置,包括但不限于:台式计算机、大型计算机、膝上型计算机、基于现场可编程门阵列(FPGA)的装置、智能电话、个人数字助理(PDA)、身佩式或插入计算机、嵌入式装置类型计算机、基于专用集成电路(ASIC)的装置。

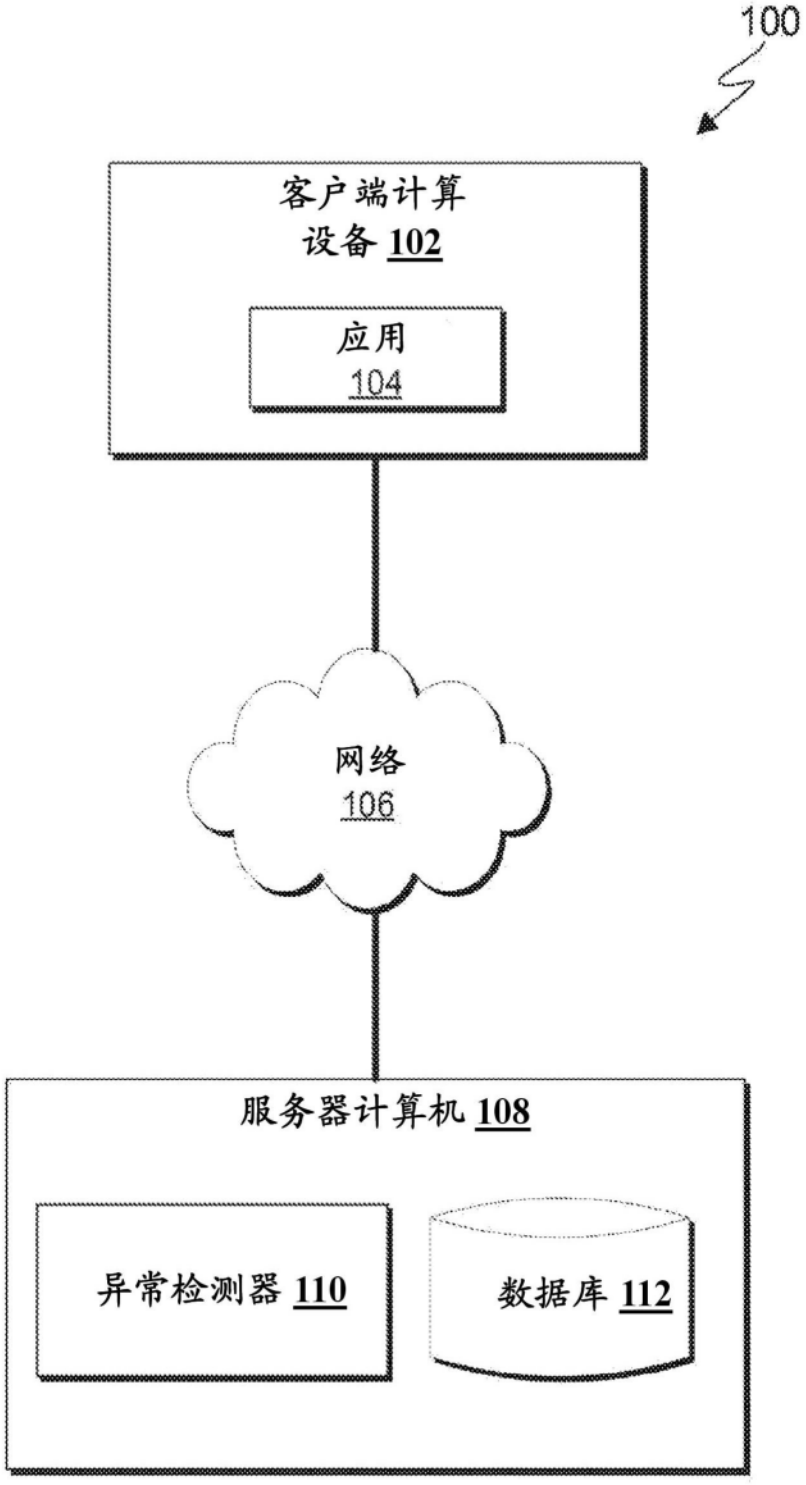


图1

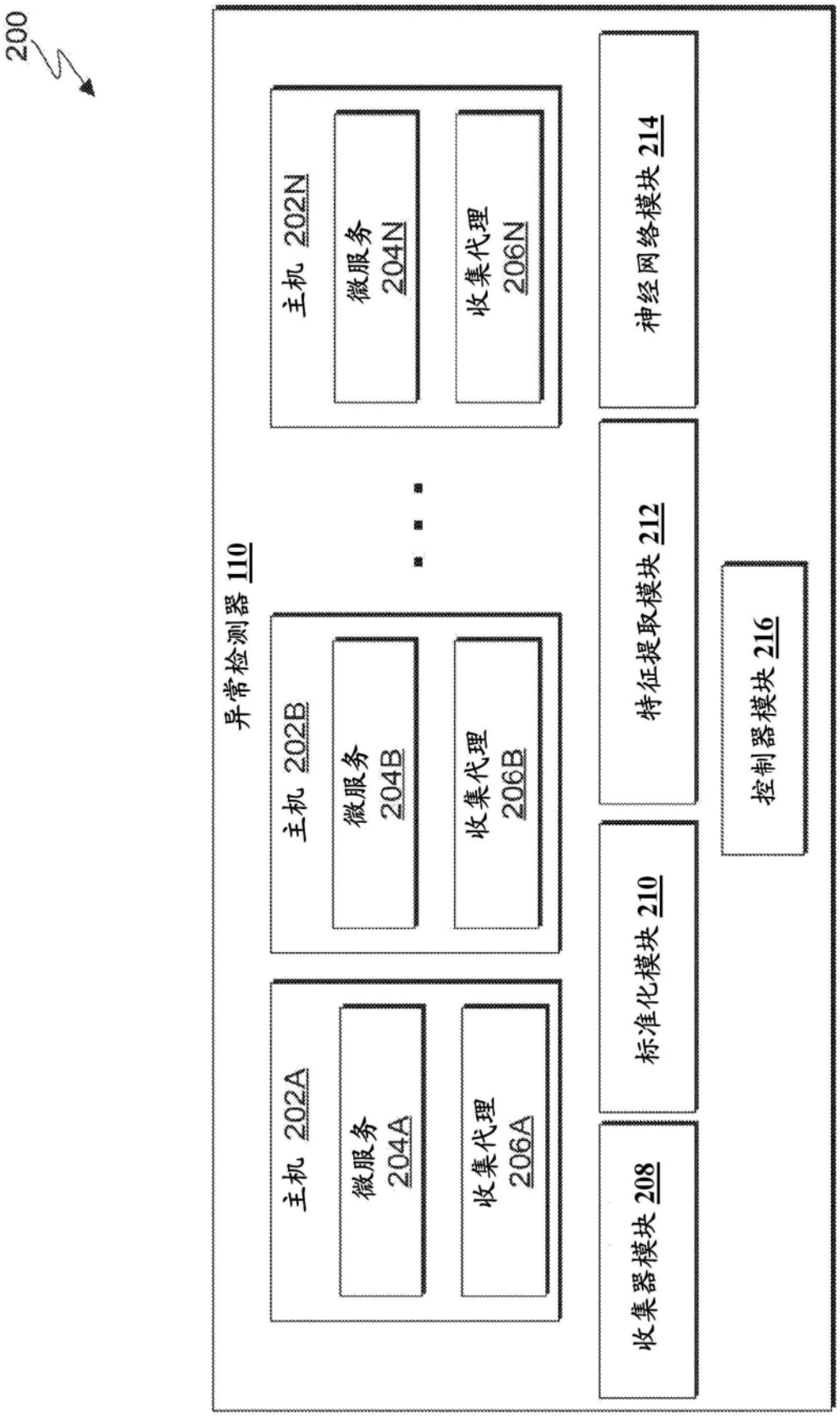

200
异常检测器 110
主机 202A
微服务 204A
收集代理 206A
主机 202B
微服务 204B
收集代理 206B
主机 202N
微服务 204N
收集代理 206N
收集器模块 208
标准化模块 210
特征提取模块 212
神经网络模块 214
控制器模块 216

图2

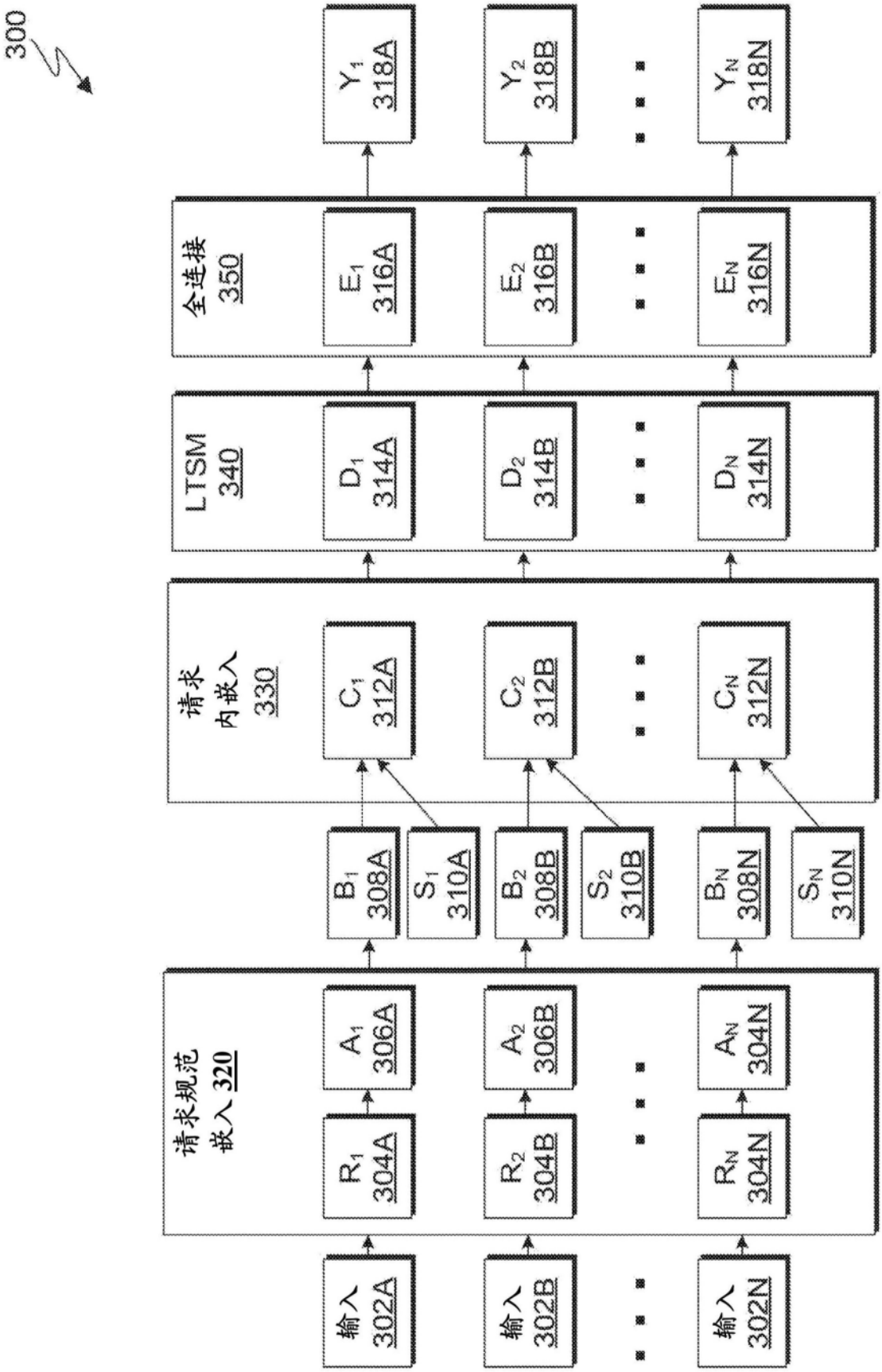

300
输入 302A
输入 302B
输入 302N
请求规范嵌入 320
R1 304A
R2 304B
RN 304N
A1 306A
A2 306B
AN 304N
请求内嵌入 330
B1 308A
B2 308B
BN 308N
S1 310A
S2 310B
SN 310N
C1 312A
C2 312B
CN 312N
LTSM 340
D1 314A
D2 314B
DN 314N
全连接 350
E1 316A
E2 316B
EN 316N
Y1 318A
Y2 318B
YN 318N

图3

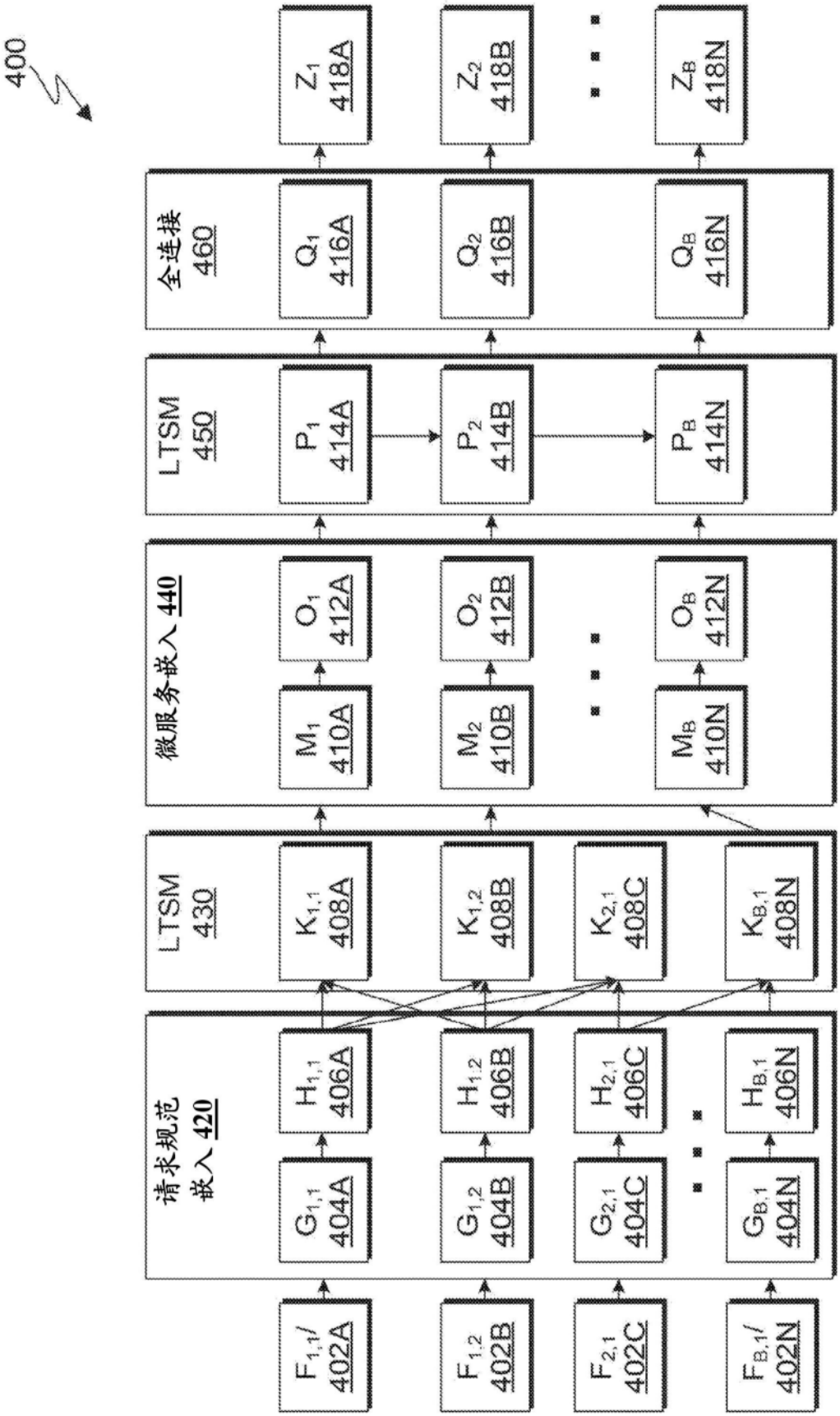

400
请求规范嵌入 420
LTSM 430
微服务嵌入 440
LTSM 450
全连接 460
F1,1/ 402A
F1,2 402B
F2,1 402C
FB,1/ 402N
G1,1 404A
G1,2 404B
G2,1 404C
GB,1 404N
H1,1 406A
H1,2 406B
H2,1 406C
HB,1 406N
K1,1 408A
K1,2 408B
K2,1 408C
KB,1 408N
M1 410A
M2 410B
MB 410N
O1 412A
O2 412B
OB 412N
P1 414A
P2 414B
PB 414N
Q1 416A
Q2 416B
QB 416N
Z1 418A
Z2 418B
ZB 418N

图4

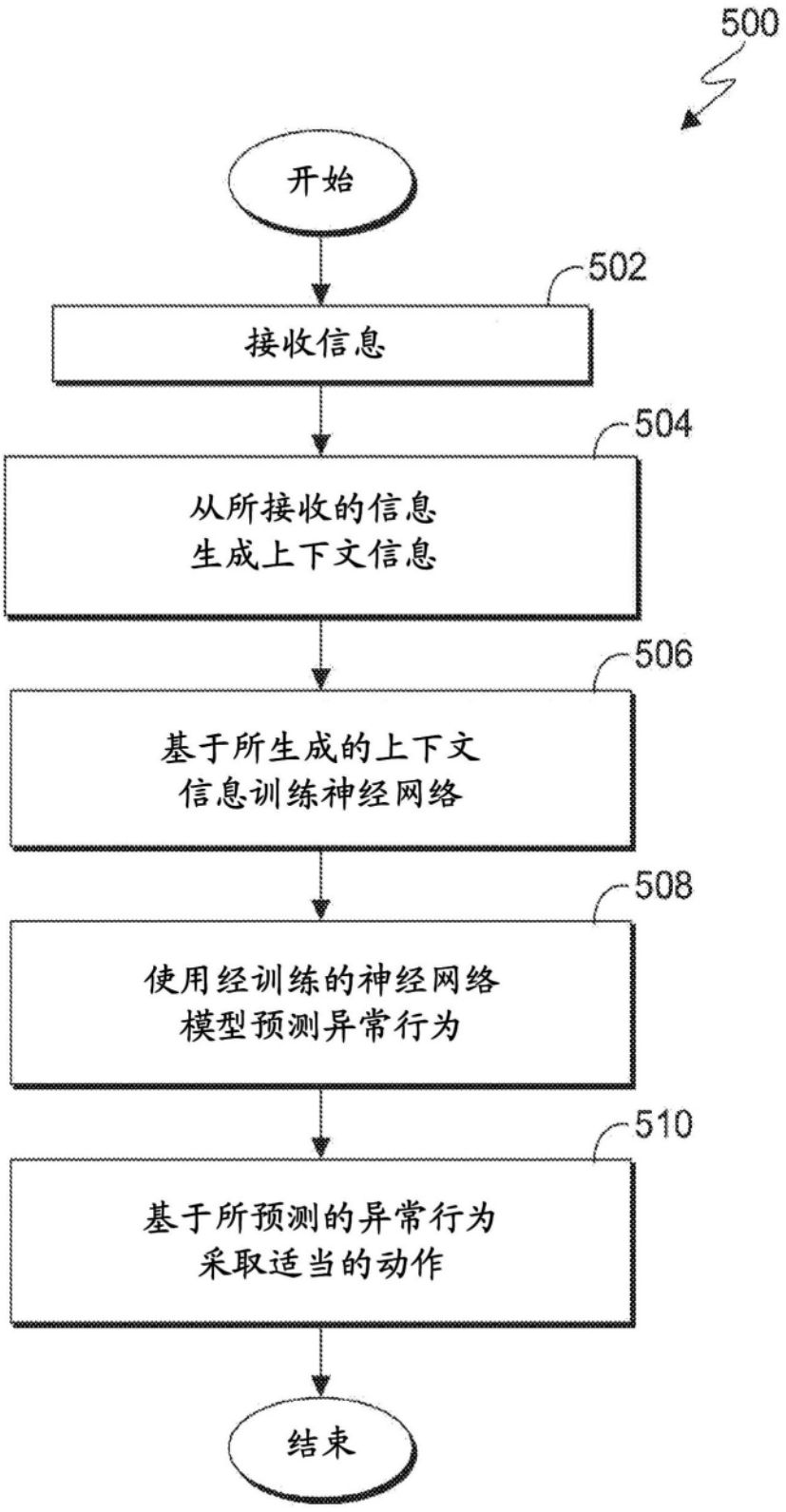

500
开始
502
接收信息
504
从所接收的信息
生成上下文信息
506
基于所生成的上下文
信息训练神经网络
508
使用经训练的神经网络
模型预测异常行为
510
基于所预测的异常行为
采取适当的动作
结束

图5

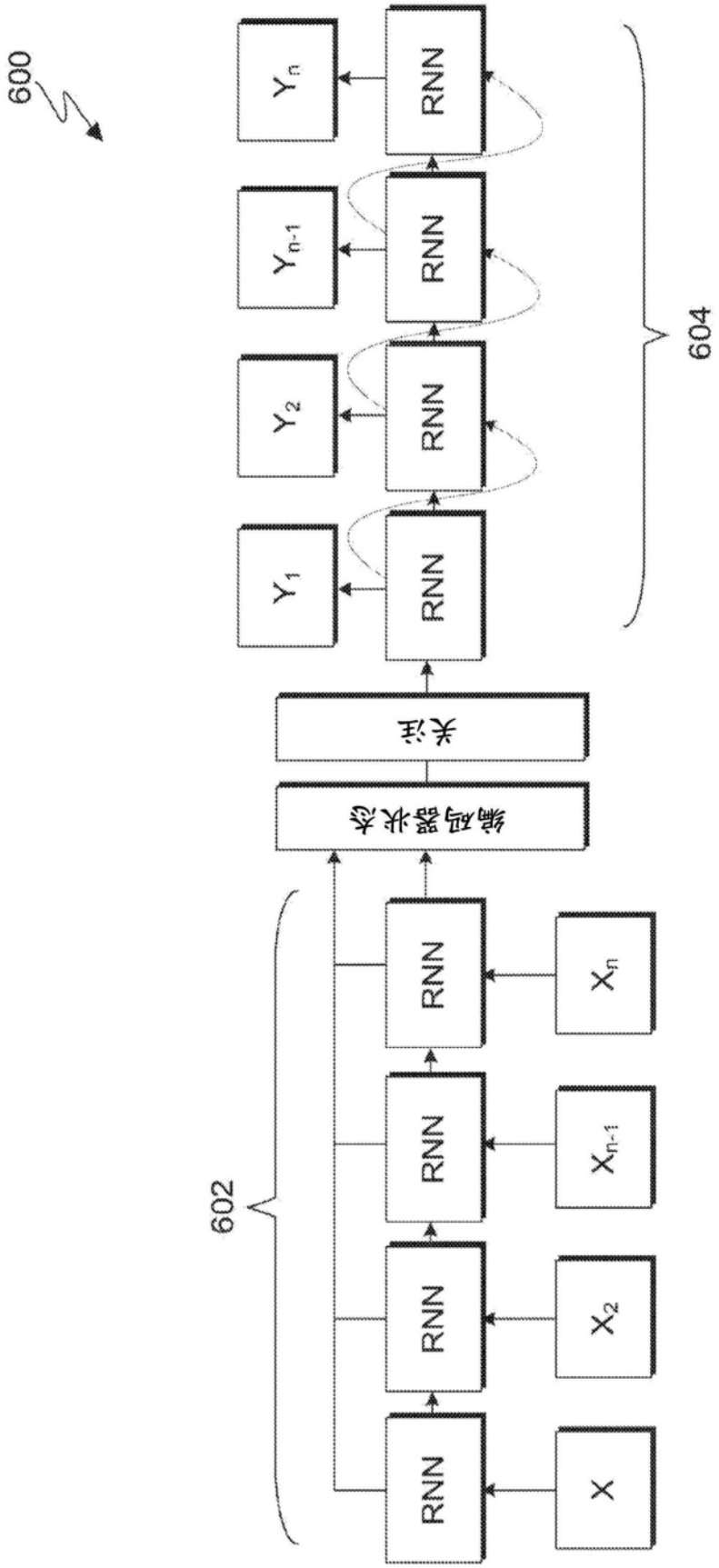


图6


```
//针对请求
完成一次
receive_request()
{
        request_spe
c()
}
function_call()
{
        start_trace()
        // 业务
逻辑
        end_trace()
}
```

图7A


```
//针对实例
完成一次
agent_init()
{
        static_info()
}
receive_trace()
{
        resource_info
()
}
```

图7B

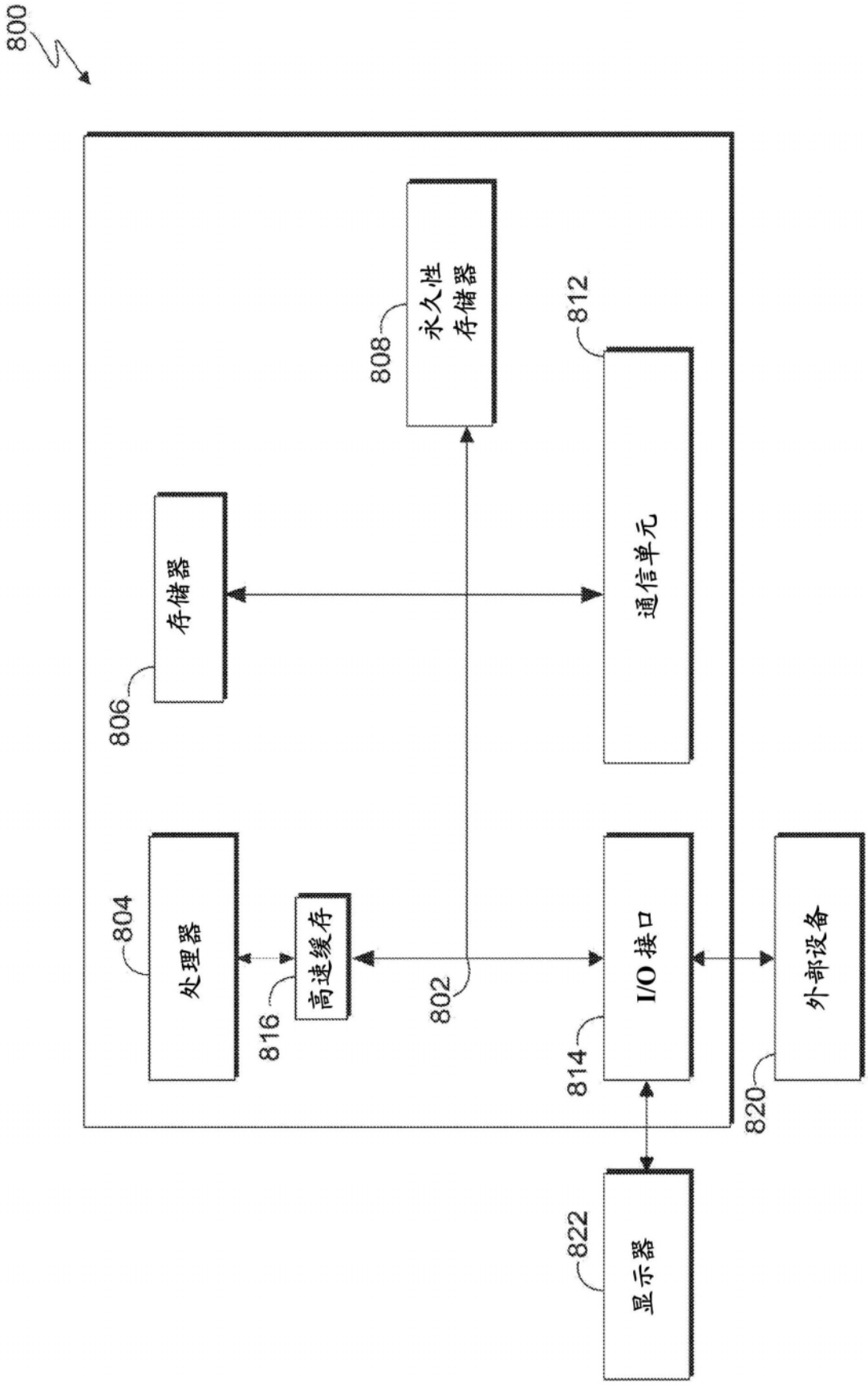

800
永久性存储器
808
812
存储器
806
通信单元
处理器
804
高速缓存
816
802
I/O 接口
814
外部设备
820
显示器
822

图8