(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 111263933 A
(43)申请公布日 2020.06.09

(21)申请号 201880069160.8

(22)申请日 2018.09.28

(30)优先权数据
62/566,351 2017.09.30 US

(85)PCT国际申请进入国家阶段日
2020.04.23

(86)PCT国际申请的申请数据
PCT/US2018/053628 2018.09.28

(87)PCT国际申请的公布数据
WO2019/068037 EN 2019.04.04

(71)申请人 甲骨文国际公司
地址 美国加利福尼亚

(72)发明人 C·卡尔达图 B·舒赫

(74)专利代理机构 中国国际贸易促进委员会专利商标事务所 11038
代理人 刘玉洁

(51)Int.Cl.
G06F 11/36(2006.01)
G06F 8/60(2006.01)
G06F 9/50(2006.01)

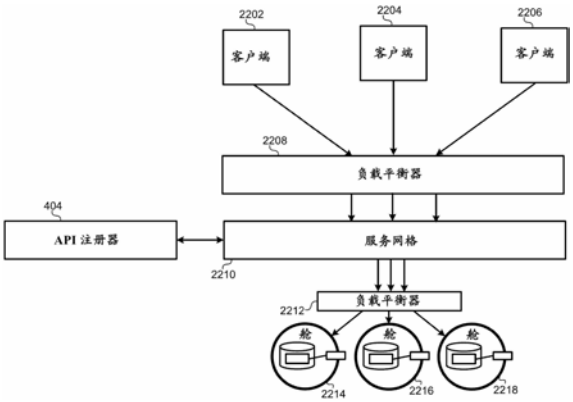
权利要求书2页 说明书32页 附图35页

(54)发明名称

部署的容器平台中的实时调试实例

(57)摘要

一种方法可以包括接收对在容器环境处的服务的请求。容器环境可以包括服务网格和封装在多个容器中的多个服务。服务可以被封装在第一一个或多个容器中。该方法还可以包括确定请求应当被路由到服务的调试实例;以及实例化服务的调试实例。调试实例可以被封装在第二一个或多个容器中,并且可以包括实现服务的代码和一个或多个调试实用程序。该方法可以附加地包括通过服务网格将请求路由到调试实例。



CN 111263933 A

1.一种为容器环境中的容器化的服务提供运行时调试的方法,所述方法包括:

接收对在容器环境处的服务的请求,其中:

所述容器环境包括服务网格和封装在多个容器中的多个服务;以及

所述服务被封装在第一一个或多个容器中;

确定所述请求应当被路由到所述服务的调试实例;

实例化所述服务的所述调试实例,其中所述调试实例被封装在第二一个或多个容器中并且包括:

实现所述服务的代码;以及

一个或多个调试实用程序;

通过所述服务网格将所述请求路由到所述调试实例。

2.如权利要求1所述的方法,其中所述第一一个或多个容器被组织成容器舱。

3.如权利要求1所述的方法,其中所述容器环境包括编排的容器平台,所述编排的容器平台包括容器调度器。

4.如权利要求3所述的方法,其中所述容器调度器使所述服务的所述调试实例被实例化。

5.如权利要求1所述的方法,其中所述容器环境包括使得所述服务的所述调试实例被实例化的应用编程接口API注册器。

6.如权利要求5所述的方法,其中所述API注册器接收针对所述服务的所述调试实例的注册,并通过API函数调用使所述服务的所述调试实例的HTTP端点可用。

7.如权利要求5所述的方法,其中所述API注册器接收针对所述服务的注册,所述注册包括指示应当实例化所述服务的调试实例的特性。

8.一种包括指令的非暂态计算机可读介质,所述指令在由一个或多个处理器执行时使所述一个或多个处理器执行操作,所述操作包括:

接收对在容器环境处的服务的请求,其中:

所述容器环境包括服务网格和封装在多个容器中的多个服务;以及

所述服务被封装在第一一个或多个容器中;

确定所述请求应当被路由到所述服务的调试实例;

实例化所述服务的所述调试实例,其中所述调试实例被封装在第二一个或多个容器中并且包括:

实现所述服务的代码;以及

一个或多个调试实用程序;

通过所述服务网格将所述请求路由到所述调试实例。

9.如权利要求8所述的非暂态计算机可读介质,其中所述服务被封装在单个容器中。

10.如权利要求9所述的非暂态计算机可读介质,其中所述单个容器还包括所述一个或多个调试实用程序。

11.如权利要求9所述的非暂态计算机可读介质,其中所述一个或多个调试实用程序被封装在除了所述单个容器之外的至少一个容器中。

12.如权利要求9所述的非暂态计算机可读介质,其中所述一个或多个调试实用程序包括用于监视存储器使用情况或处理器使用情况的进程。

13.如权利要求9所述的非暂态计算机可读介质,其中所述一个或多个调试实用程序包括调试守护进程。

14.如权利要求9所述的非暂态计算机可读介质,其中实现所述服务的代码包括所述服务的所述调试版本。

15.一种系统,包括:

一个或多个处理器;以及

包括指令的一个或多个存储器设备,所述指令在由所述一个或多个处理器执行时使所述一个或多个处理器执行操作,所述操作包括:

接收对在容器环境处的服务的请求,其中:

所述容器环境包括服务网格和封装在多个容器中的多个服务;以及

所述服务被封装在第一一个或多个容器中;

确定所述请求应当被路由到所述服务的调试实例;

实例化所述服务的所述调试实例,其中所述调试实例被封装在第二一个或多个容器中并且包括:

实现所述服务的代码;以及

一个或多个调试实用程序;

通过所述服务网格将所述请求路由到所述调试实例。

16.如权利要求15所述的系统,其中所述服务的所述调试实例在接收所述请求之前被实例化。

17.如权利要求15所述的系统,其中所述服务的所述调试实例响应于接收到所述请求而被实例化。

18.如权利要求15所述的系统,其中确定所述请求应当被路由到所述服务的所述调试实例包括识别所述请求的源。

19.如权利要求15所述的系统,其中确定所述请求应当被路由到所述服务的所述调试实例包括识别所述请求中将所述请求指定为调试请求的报头。

20.如权利要求15所述的系统,其中,所述请求被转发到所述服务的所述调试实例,而不中断其它请求到所述服务的路由。

部署的容器平台中的实时调试实例

[0001] 相关申请的交叉引用

[0002] 本申请要求于2017年9月30日提交的美国临时申请No.62/566,351的权益,该申请通过引用并入本文。本申请还涉及与本申请在同一天提交的以下共同受让的申请,这些申请中的每个申请也通过引用并入本文:

[0003] • 于2018年9月__日提交的美国专利申请No.__/___,___,题为“API REGISTRY IN A CONTAINER PLATFORM FOR AUTOMATICALLY GENERATING CLIENT CODE LIBRARIES”(代理人案号088325-1090745);

[0004] • 于2018年9月__日提交的美国专利申请No.__/___,___,题为“API REGISTRY IN A CONTAINER PLATFORM PROVIDING PROPERTY-BASED API FUNCTIONALITY”(代理人案号088325-1090746);

[0005] • 于2018年9月__日提交的美国专利申请No.__/___,___,题为“DYNAMIC NODE REBALANCING BETWEEN CONTAINER PLATFORMS”(代理人案号088325-1090747);

[0006] • 于2018年9月__日提交的美国专利申请No.__/___,___,题为“OPTIMIZING REDEPLOYMENT OF FUNCTIONS AND SERVICES ACROSS MULTIPLE CONTAINER PLATFORMS AND INSTALLATIONS”(代理人案号088325-1090748)。

背景技术

[0007] 概括地说,任何形式的容器都表示包装和与信息交互的标准化方法。容器可以彼此隔离并可以并行使用,而没有交叉污染的风险。在现代软件世界中,“容器”一词具有特定的含义。软件容器,诸如**Docker®**容器,是在逻辑上封装并定义软件片段的软件构造。要封装在容器中的最常见的软件类型是应用、服务或微服务。现代容器还包括应用/服务操作所需的所有软件支持,诸如操作系统、库、存储卷、配置文件、应用二进制文件以及将在典型计算环境中找到的技术堆栈的其它部分。然后,可以使用这个容器环境创建多个容器,每个容器在任何环境中都运行其自己的服务。容器可以部署在生产数据中心、内部部署的(on-premises)数据中心、云计算平台等中,而无需进行任何改变。在云上启动(spin up)容器与在本地工作站上启动容器相同。

[0008] 现代的面向服务的体系架构和云计算平台将大任务分解成许多小的具体任务。容器可以被实例化以专注于各个具体任务,然后多个容器可以协同工作以实现复杂的应用。这可以被称为微服务体系架构,并且每个容器可以使用可以独立升级的不同版本的编程语言和库。与将要对更大、更整体的体系架构进行的改变相比,容器内处理的隔离性质允许它们以很小的工作或风险进行升级或更换。虽然虚拟机可以被用于运行容器平台,但是容器平台在运行这种微服务体系架构方面比传统虚拟机效率高得多。

发明内容

[0009] 在一些实施例中,一种为容器环境中的容器化的服务提供运行时调试的方法可以包括接收对在容器环境处的服务的请求。容器环境可以包括服务网格和封装在多个容器中

的多个服务。服务可以被封装在第一一个或多个容器中。该方法还可以包括确定该请求应当被路由到服务的调试实例;以及实例化服务的调试实例。调试实例可以被封装在第二一个或多个容器中,并且可以包括实现服务的代码和一个或多个调试实用程序。该方法可以附加地包括通过服务网格将请求路由到调试实例。

[0010] 在一些实施例中,一种非暂态计算机可读介质可以包括指令,这些指令在由一个或多个处理器执行时使一个或多个处理器执行包括接收对在容器环境处的服务的请求的操作。容器环境可以包括服务网格和封装在多个容器中的多个服务。服务可以被封装在第一一个或多个容器中。操作还可以包括确定该请求应当被路由到服务的调试实例;以及实例化服务的调试实例。调试实例可以被封装在第二一个或多个容器中,并且可以包括实现服务的代码和一个或多个调试实用程序。操作可以附加地包括通过服务网格将请求路由到调试实例。

[0011] 在一些实施例中,一种系统可以包括一个或多个处理器以及包括指令的一个或多个存储器设备,这些指令在由一个或多个处理器执行时使一个或多个处理器执行包括接收对在容器环境处的服务的请求的操作。容器环境可以包括服务网格和封装在多个容器中的多个服务。服务可以被封装在第一一个或多个容器中。操作还可以包括确定该请求应当被路由到服务的调试实例;以及实例化服务的调试实例。调试实例可以被封装在第二一个或多个容器中,并且可以包括实现服务的代码和一个或多个调试实用程序。操作可以附加地包括通过服务网格将请求路由到调试实例。

[0012] 在任何实施例中,以下特征中的任何一个或全部可以以任何组合被包括并且没有限制。可以将第一一个或多个容器组织到容器舱中。容器环境可以包括包含容器调度器的经编排的容器平台。容器调度器可以使得服务的调试实例被实例化。容器环境可以包括使得服务的调试实例被实例化的应用编程接口(API)注册器。API注册器可以接收服务的调试实例的注册,并通过API函数调用使服务的调试实例的HTTP端点可用。API注册器可以接收服务的注册,该注册包括指示服务的调试实例应当被实例化的特性。服务可以被封装在单个容器中。单个容器还可以包括一个或多个调试实用程序。可以将一个或多个调试实用程序封装在除单个容器之外的至少一个容器中。一个或多个调试实用程序可以包括用于监视存储器使用情况或处理器使用情况的进程。一个或多个调试实用程序可以包括调试守护进程。实现服务的代码可以包括服务的调试版本。服务的调试实例可以在接收请求之前被实例化。服务的调试实例可以响应于接收到请求而被实例化。确定应当将请求路由到服务的调试实例可以包括识别请求的源。确定请求应当被路由到服务的调试实例可以包括识别请求中将该请求指定为调试请求的报头。可以将请求转发到服务的调试实例,而不中断其它请求到服务的路由。

附图说明

[0013] 通过参考本说明书的其余部分和附图,可以实现对本发明的本质和优点的进一步理解,其中,贯穿若干附图使用了相似的附图标记指代相似的部件。在一些情况下,子标签与附图标记相关联以表示多个相似部件中的一个部件。当在没有指定现有子标签的情况下参考附图标记时,意在指代所有这样的多个相似部件。

[0014] 图1图示了根据一些实施例的用于容器平台中的服务的开发和运行时环境的软件

结构和逻辑布置。

[0015] 图2图示了专门设计为运行本文所述实施例的专用计算机硬件系统。

[0016] 图3图示了可以特定于由本文所述的实施例中的一些实施例使用的容器平台的数据组织。

[0017] 图4图示了根据一些实施例的可以被部署到IDE和生产/运行时环境的API注册器。

[0018] 图5图示了根据一些实施例的用于在运行时与容器平台一起使用的API注册器的部署。

[0019] 图6A图示了根据一些实施例的用于部署API注册器的方法的流程图。

[0020] 图6B图示了根据一些实施例的当使用图6A中的流程图来部署API注册器时的容器平台的软件结构。

[0021] 图7A图示了根据一些实施例的用于向API注册器注册服务的方法的流程图。

[0022] 图7B图示了根据一些实施例的用于向API注册器注册API的步骤的硬件/软件示意图。

[0023] 图8图示了根据一些实施例的用于浏览和选择向API注册器注册的API的图形界面和命令行界面的示例。

[0024] 图9图示了根据一些实施例的用于使用向API注册器注册的服务及其对应功能的方法的流程图。

[0025] 图10图示了API注册器可以如何通过CreateUser()函数的图形界面接收选择。

[0026] 图11图示了根据一些实施例的由API注册器自动为服务生成的客户端库的示例。

[0027] 图12图示了根据一些实施例的适应(accommodate)服务端点与API函数之间的动态绑定的客户端库的实施例。

[0028] 图13图示了根据一些实施例的可以整理(marshal)附加数据以完成用于服务调用的输入数据集的客户端库的实施例。

[0029] 图14图示了根据一些实施例的在调用服务时可以处置(handle)重试的客户端库。

[0030] 图15A图示了根据一些实施例的向API注册器提供API特性的方法。

[0031] 图15B图示了根据一些实施例的服务可以如何向API注册器提供API特性的硬件/软件示意图。

[0032] 图16图示了根据一些实施例的硬件/软件示意图,其中API注册器使用特性来部署具有高可用性的服务。

[0033] 图17图示了根据一些实施例的通过API注册器强制实施端到端加密的特性的硬件/软件示意图。

[0034] 图18图示了根据一些实施例的用于实现服务1808的使用情况日志记录的API注册器的特性。

[0035] 图19图示了根据一些实施例的可以对服务强制实施认证协议的特性的硬件/软件示意图。

[0036] 图20图示了根据一些实施例的用于启用服务的运行时实例化的特性的硬件/软件示意图。

[0037] 图21图示了根据一些实施例的实现用于服务的速率限制功能的特性的硬件/软件示意图。

[0038] 图22图示了根据一些实施例的用于接收服务请求的云计算平台的一部分的框图。

[0039] 图23图示了根据一些实施例的封装在舱中的服务的调试版本。

[0040] 图24图示了根据一些实施例的用于服务的调试版本的替代舱。

[0041] 图25图示了根据一些实施例的用于实例化服务的调试实例的系统的框图。

[0042] 图26图示了根据一些实施例的将调试请求路由到服务的调试实例的容器平台的框图。

[0043] 图27图示了根据一些实施例的克隆请求的云计算平台的框图。

[0044] 图28图示了根据一些实施例的克隆的请求被转发到服务的调试实例的框图。

[0045] 图29图示了用于为容器环境中的容器化的服务提供运行时调试的方法的流程图。

[0046] 图30图示了用于实现实施例中的一些实施例的分布式系统的简化框图。

[0047] 图31图示了系统环境的部件的简化框图，通过该系统环境，由实施例系统的部件提供的服务可以作为云服务提供。

[0048] 图32图示了其中可以实现各种实施例的示例性计算机系统。

具体实施方式

[0049] 本文描述的是应用编程接口(API)注册器的实施例，该API注册器是允许开发人员在开发期间注册服务并使那些服务在部署期间和部署之后对其它服务可用的集成开发环境(IDE)的一部分。API注册器可以作为编排的容器平台的一部分进行部署，从而在容器平台上作为容器化的应用操作。随着服务或微服务被开发并部署到容器平台上的容器中，API注册器可以执行在容器平台内定位与可用服务对应的可用端点(例如，IP地址和端口号)的发现过程。API注册器还可以接受API定义文件的上传，该API定义文件可以被用于将原始服务端点转换成通过API注册器可用的API函数。API注册器可以动态地将发现的端点绑定到API函数，该API函数可以保持最新状态并可以对容器平台中的其它服务可用。这提供了其它服务可以静态调用的稳定端点，而API注册器管理服务端点中API函数之间的绑定的任何改变。这也简化了在容器平台中使用服务的过程。代替为HTTP调用编写代码，新服务可以仅使用API接口来访问已注册的服务。

[0050] 在一些实施例中，IDE可以为开发人员提供导航/浏览界面，以定位容器平台中可用并向API注册器注册的服务。当API注册器为正在开发的新服务创建对现有服务的调用时，API注册器可以自动生成包括与已注册的服务进行交互的所有必要功能的客户端库的集合。例如，一些实施例可以生成包括与API调用对应的成员函数的对象类。在开发期间，新服务可以简单地实例化这些对象和/或使用其成员函数来调用对应的API。客户端库中的代码管理调用服务与已注册的服务的端点之间的直接连接，并且可以包括处置这种交互所必需的所有功能的代码。例如，自动生成的客户端库可以包括：用于将来自API调用的参数打包和格式化为对服务端点的HTTP调用的代码、用于整理数据以完成用于该调用的参数集的代码、用于将信息打包到兼容分组(JSON、XML等)中的代码、用于接收和解析结果分组的代码、用于处置重试和错误状况的代码等。从调用服务的角度来看，用于处置所有这种功能的代码由API注册器自动生成，并且因此将服务调用的细节抽象并封装到客户端库对象中。调用服务所需要的只是执行由API注册器创建的客户端库对象的成员函数。

[0051] 在一些实施例中，API注册器也可以接受可以定义已注册的服务的运行时执行的

特性集合的上传。这个特性集合可以在开发期间与API定义文件一起上传。这些特性可以定义运行时特点，诸如端到端加密、使用/日志记录要求、用户认证、按需服务实例化、用于高可用性的多个服务部署实例、速率/使用情况限制以及其它运行时特点。API注册器可以通过在开发期间、在部署期间以及在运行时期间与容器环境进行交互来确保满足这些特性。在开发期间，用于调用服务的自动生成的客户端库可以包括执行这些特性所需的代码，诸如加密代码、使用情况日志记录代码和/或与用户认证服务的交互。当部署已注册的服务时，API注册器可以指示容器平台实例化服务的多个实例和/或其它负载平衡模块，以确保服务在运行时期间具有高可靠性。在运行时期间当服务被调用时，API注册器可以使服务被实例化以进行按需实例化、限制可以进行的API调用的次数以调节使用情况，以及执行其它运行时功能。

[0052] 图1图示了根据一些实施例的用于容器平台中的服务的开发和运行时环境的软件结构和逻辑布置。这些环境可以包括IDE 102，其可以被用于开发要部署在容器平台上的服务和微服务。IDE是软件套件，其合并以及提供服务开发人员可以用来编写和测试新服务的所有基本工具。IDE 102可以包括具有图形用户界面(GUI)、代码完成功能以及导航/浏览界面的源代码编辑器106，其允许开发人员编写、导航、集成和可视化源代码编写过程。IDE 102还可以包括调试器110，该调试器110包括变量接口、立即(immediate)变量接口、表达式评估接口、存储器内容接口、断点可视化和功能以及其它调试功能。IDE 102还可以包括用于编译和运行经编译的机器代码或经解释的字节代码的编译器和/或解释器108。编译器/解释器108可以包括允许开发人员使用/生成另一个构建自动化构造的makefile(制作文件)的构建工具。IDE 102的一些实施例可以包括代码库112，其包括可以链接到正在开发的服务中并且跨多个开发重用的通用代码函数、对象、接口和/或其它结构。

[0053] 可以在IDE 102内开发服务并对服务进行彻底的测试，直到准备好进行部署为止。然后可以将服务部署到生产/部署环境104。生产/开发环境104可以包括许多不同的硬件和/或软件结构，包括专用硬件、虚拟机和容器化的平台。在本公开之前，当将服务114部署到生产/部署环境104中时，服务114将不再具有对在IDE 102中使用的工具中的许多工具的运行时访问。服务114在生产/开发环境104中运行所需的任何功能都需要从代码库112中打包，并与服务114一起部署到生产/部署环境104中。此外，通常服务114将被部署为没有调试器110的任何功能或来自源代码编辑器106的源代码的副本。本质上，服务114将被部署到生产/部署环境104，其具有运行时操作所需的所有功能，但是将被剥离仅在开发期间使用的信息。

[0054] 图2图示了专门设计用于运行本文所述的实施例的专用计算机硬件系统。举例来说，服务114可以被部署到基础设施即服务(IaaS)云计算环境202中。这是云计算的形式，云计算通过网络提供虚拟化或共享的计算资源。IaaS云计算环境202还可以包括被布置为软件即服务(SaaS)和/或平台即服务(PaaS)体系架构的其它云计算环境，或与其它云计算环境耦合。在这种环境中，云提供商可以托管传统上存在于内部部署的数据中心中的硬件和/或软件部件的基础设施。这种硬件可以包括服务器、存储装置、联网硬件、盘阵列、软件库和虚拟化实用程序(诸如管理程序层)。IaaS环境202可以由商业源(诸如Oracle®或其它公共可用的云平台)提供。IaaS环境202也可以使用硬件和软件的私有基础设施被部署为私有云。

[0055] 不管云环境的类型如何，都可以将服务114部署到多种不同类型的硬件/软件系统上。例如，服务114可以被部署到专用硬件206。专用硬件206可以包括专门指派给服务114的硬件资源，诸如服务器、盘、操作系统、软件包等。例如，特定服务器可以被分配为处置流入和流出服务114的业务。

[0056] 在另一个示例中，服务114可以被部署到作为一个或多个虚拟机208操作的硬件/软件。虚拟机是提供专用计算机硬件206的功能的计算机系统的模拟。但是，作为专用于特定功能的代替，物理硬件可以由多个不同的虚拟机共享。每个虚拟机可以提供执行所需的所有功能，包括完整的操作系统。这允许具有不同操作系统的虚拟机在同一个物理硬件上运行并允许多个服务共享单个硬件。

[0057] 在另一个示例中，服务114可以被部署到容器平台210。容器平台在多个重要方面与虚拟机208不同。首先，容器平台210将各个服务打包到容器中，如下面在图3中更详细描述的。每个容器共享主机操作系统内核，并且它们还共享二进制文件、库和其它只读组件。这使容器非常轻量级-常常只有几兆字节的大小。此外，轻量级容器非常高效，起动仅需几秒钟，而引导虚拟机则需要几分钟。容器还通过共享可以为容器平台210中的整个容器集合一起维护的操作系统和其它库来减少管理开销。尽管容器共享同一个操作系统，但它们提供隔离的平台，因为操作系统为隔离提供虚拟存储器支持。容器技术可以包括**Docker®**容器、Linux **Libcontainer®**、开放式容器计划(OCI)、**Kubernetes®**、CoeOS、**Apache®** Mesos等。可以将这些容器部署到容器编排平台，其在本文中可以被简称为“容器平台”210。容器平台管理已部署的软件容器的自动化布置、协调和管理。容器平台210可以提供服务发现、负载平衡、健康状况检查、多个部署等。容器平台210可以由运行在节点和舱中组织的容器的任何公共可用的容器平台(诸如Kubernetes)来实现。

[0058] 不管在其上部署了服务114的平台206、208、210，平台206、208、210中的每一个都可以提供为调用服务114提供公共访问的服务端点212、214、216。一般而言，这些端点可以通过HTTP调用进行访问并与IP地址和端口号相关联。通过连接到正确的IP地址和端口号，当其它服务公开可用时，这些其它服务可以调用部署到平台206、208、210中的任何一个的服务。每个服务(诸如服务114)可以包括其自己的用于调用该服务的专有格式和数据要求。类似地，每个服务可以返回在格式和数据类型方面特定于该服务114的结果。除了特定于服务的要求之外，特定的部署平台206、208、210还可以包括与服务114交互的附加要求，诸如编程语言、需要符合以便与服务正确交互的数据包格式(JSON、XML等)等。

[0059] 虽然以上示例允许将服务114部署到所述平台206、208、210中的任何一个，但是本文所述的实施例是专门为上述容器平台210设计的。因此，可以将专门记载为要部署在“容器平台”中的实施例与专门记载为要部署在虚拟机平台中、部署在服务器或专用硬件平台上或一般而言部署在IaaS环境中的其它实施例区分开。

[0060] 图3图示了可以特定于由本文所述的实施例中的一些实施例使用的容器平台210的数据组织。一般而言，服务到容器平台的任何部署都将被部署到舱304、306。舱是表示一组一个或多个应用容器(例如Docker或rkt)的抽象。舱还可以包括通常可用于该舱内的所有容器的一些共享资源。例如，舱304包括容器310和容器312。舱304还包括共享资源308。资源可以包括存储卷或关于容器如何在舱304内运行或连接的其它信息。舱304可以对包含相

对紧密耦合的不同服务容器310、312的专用逻辑主机进行建模。例如，容器310中的服务326可以利用资源308并调用容器312中的服务320。服务320还可以调用服务322，服务322进而调用服务324，每个服务都部署到容器312。服务324的输出可以被提供给网络IP地址和端口318，这是由舱304共享的另一个公共资源。因此，服务320、322、324、326全部与共享资源308一起工作以提供单个服务，该单个服务可以由在其它容器中运行的服务通过IP地址和端口号318访问。服务还可以由容器平台外部的计算机系统(诸如工作站、膝上型计算机、智能电话或不属于容器平台或IaaS环境的一部分的其它计算设备)通过IP地址和端口318访问。

[0061] 在最简单的部署中，每个容器可以包括单个服务，并且每个舱可以包括封装服务的单个容器。例如，舱306仅包括具有单个服务328的单个容器314。可以通过舱306的IP地址和端口号316访问该单个服务。通常，当将服务部署到容器平台时，将实例化容器和舱以保持服务。可以将许多不同的舱部署到容器节点302。一般而言，舱在节点内运行。节点表示容器平台中的工作者机器(虚拟的或者物理的)。每个节点由“主设备”管理，该“主设备”自动处置在每个节点内调度舱。每个节点可以运行负责主设备和该节点之间的通信并负责管理该节点表示的机器上的容器中的舱的过程。每个节点还可以包括负责从注册器中拉取容器映像、对容器进行解包并运行服务的容器运行时。

[0062] 图4图示了根据一些实施例的可以被部署到IDE 102和生产/运行时环境104的API注册器404。如上所述，存在技术问题，其中当将服务114从IDE 102部署到生产/部署环境104时，服务114失去对仅仅在IDE 102中可用的信息的运行时访问。当API注册器404在生产/开发环境104中在运行时期间部署和操作时，API注册器404可由服务114访问。通过在开发期间向API注册器404注册服务并将API定义和/或API特性提供给API注册器404，API注册器404克服了开发功能与运行时功能隔离的以前的技术问题。定义API的信息可以由IDE 102中正在开发的新服务以及已部署到生产/部署环境104的服务使用。在完成这个注册过程之后，服务114可以使用在运行时期间访问API注册器404的客户端库进行操作，以确保API函数正确绑定到对应服务的当前IP地址和端口号。API注册器404表示专门设计用于解决这些技术问题的新的数据结构和处理单元。

[0063] 现有技术中存在的另一个技术问题是在将服务特性部署到生产/开发环境104时实现服务特性。例如，如果要部署具有高可用性的服务，那么开发人员将需要构建专门实例化容器平台中的服务的多个实例并以服务始终可用的方式平衡业务的容器部署文件。服务开发人员并不总是拥有这种专业知识，他们也常常无法管理他们的服务的部署。如下所述，API注册器404允许服务简单地选择特性(诸如高可用性)，特性然后可以由API注册器404自动实现。这种技术解决方案是可行的，因为API注册器404消除了IDE 102与生产/部署环境104之间的隔阂。

[0064] 图5图示了根据一些实施例的在运行时与容器平台210一起使用的API注册器404的部署。由API注册器404提供的技术解决方案和对现有技术的改进之一是维护用于服务调用的稳定端点，以及用于访问服务调用的简化和自动代码生成。在本公开之前，服务之间的调用是例如使用对IP地址和端口号的HTTP调用的点对点连接。随着服务在容器平台210中被更新、替换、重定位和重新部署，IP地址和端口号可能频繁地改变。这要求调用更新后的服务的所有服务都必须在调用该服务的实际代码中更新它们的IP地址和端口号。API注册器404通过在服务的IP地址和端口号与通过API注册器可用的API函数之间提供动态绑定来

解决这个技术问题。由API注册器404自动生成的客户端库可以包括访问API注册器404以检索和/或核实用于特定服务的当前IP地址和端口号的功能。因此，连接到第二服务的第一服务仅需要执行一次客户端库生成，以提供到第二服务的生命周期内稳定的连接。

[0065] API注册器404解决的另一个技术问题是客户端库的自动生成。在本公开之前，第一服务访问第二服务要求开发人员编写用于访问第二服务的自定义代码。因为这种代码可能随时间改变，所以第一服务与第二服务之间的不兼容性可能增加，这要求对这两个服务进行更新。API注册器404通过上传用于自动生成用于调用服务的客户端库的API定义文件来解决这个技术问题。因此，服务可以具体指定任何其它服务中的调用代码应当如何操作，从而保证兼容性。这些客户端库还大大地简化和封装了用于调用服务的代码。如下所述，可以用特定于调用服务的语言(例如，Java、C#等)的简单的成员函数调用来替换使用IP地址和端口号的复杂HTTP调用。这允许调用服务从API注册器404中选择API函数，并且可以将实现该函数的代码作为客户端库下载到调用服务。

[0066] 图6A图示了根据一些实施例的用于部署API注册器404的方法的流程图。该方法可以包括将API注册器服务部署到容器环境(601)。可以将API注册器实现为在容器内的容器环境中操作的服务。因此，在将服务部署到容器环境中之后，API注册器可以主动运行，使得它可以在运行时被访问。API注册器还可以链接到上述现有的IDE。该方法还可以包括在容器平台中发现用于可用服务的端口(603)。在将服务部署到容器平台时，API注册器可以启动发现过程，该发现过程顺序地遍历部署到容器平台的每个服务。对于每个服务，API注册器都可以检测并记录IP地址和端口号。通过这个过程发现的IP地址和端口号的列表可以被存储在数据结构中，诸如存储在与API注册器相关联的表中。每个IP地址和端口号还可以与服务的名称或在容器平台上唯一识别服务的其它标识符一起存储。图6A中的流程图中所示的这些初始步骤为API注册器提供了开始在容器平台的运行时环境中操作并可供IDE中正在开发的服务使用的起点。

[0067] 图6B图示了根据一些实施例的当使用图6A中的流程图来部署API注册器时的容器平台210的软件结构。如上所述，API注册器404可以被部署到容器平台210中的容器620。容器620可以在一个或多个舱内以及在节点内操作，如上文在图3中所述。可以使API注册器404专用于容器平台210中的其它容器中的任何一个。在一些实施例中，也可以使API注册器404公开可用于不是容器平台210的一部分的其它设备。作为容器化的服务，API注册器404可以具有可用于其它服务的IP地址和端口号。但是，API注册器404的IP地址和端口号将仅由在客户端库中自动生成的代码使用，因此一些实施例不需要公布API注册器404的IP地址和端口号。代替地，IDE本身中的客户端库可以维护API注册器404的IP地址和端口号的最新列表，使得可以在其它服务的开发、部署和运行时期间与它联系。

[0068] 在将API注册器404部署到容器620之后，API注册器404可以执行发现过程。发现过程可以使用容器平台中节点的目录列表来利用IP地址和端口号识别实现服务的舱。API注册器404然后可以访问唯一标识符(诸如每个可用服务的号码或名称)，并且将具有每个IP地址和端口号的标识符存储在容器平台210中。可以周期性地执行这个发现过程，以检测添加到容器平台210的新服务，以及识别从容器平台210移除的现有服务。如下所述，这个发现过程还可以被用于检测现有服务的IP地址和端口号何时改变。例如，API注册器404可以发现具有端点602、604、606、608的服务。在下面描述的过程中，API注册器404可以将这些端点

602、604、606、608中的每一个绑定到向API注册器404注册的API函数。在这个初始发现之后的某个点，当与端点602相关联的服务被替换、更新或修订时，端点602的IP地址和/或端口号可以改变。API注册器404可以检测对端点602的这种改变，并更新到由API注册器44提供的现有API函数的绑定。

[0069] 类似地，API注册器404可以使用发现过程来检测何时端点不再可用，然后移除与服务相关联的API函数。在一些实施例中，当服务已经向API注册器404注册，但是对应的API函数当前未绑定到有效端点时，API注册器404可以向调用对应API函数的任何服务提供虚拟(mock)响应。例如，如果已经为与端点604对应的服务注册了API，但是端点604当前不可用，那么API注册器404可以拦截对端点604的调用并提供默认或伪数据作为响应。这允许调用与端点604相关联的服务的服务在不“断开”与这个特定服务的连接的情况下维护功能和/或继续设计过程。虚拟/测试数据场景将在下面更详细地描述。

[0070] 图7A图示了根据一些实施例的用于向API注册器404注册服务的方法的流程图。该方法可以包括接收API定义的上传(701)。API定义可以以数据分组、文件或到信息储存库的链接的形式提供。API定义可以包括可以被用于识别和定义应当绑定到与服务相关联的端点的API函数的任何信息。例如，API定义的一些实施例可以包括以下数据：服务名称或其它唯一标识符；与服务端点和调用对应的函数名称；调用服务所需的数据输入以及对应的描述和数据类型；结果数据格式和数据类型；当前的IP地址和/或端口号；描述将与端点相关联的API函数的功能的文档；在虚拟/测试场景期间应当返回的默认或伪数据值；以及可以被API注册器404用于将端点接收的HTTP请求翻译成使用类数据对象的API函数调用的客户端库的任何其它信息。

[0071] 该方法还可以包括基于上传的API定义创建对应的API函数(703)。这些API函数可以基于API定义自动生成。服务的每个端点可以与多个不同的API函数相关联。例如，实现RESTful接口的端点可以在相同的IP地址和端口号处接收对POST、GET、PUT和DELETE函数的HTTP调用。例如，这可能导致不同的API函数。例如，如果接口表示用户的列表，那么这个列表可以与至少四个不同的API函数对应，诸如GetUser()、AddUser()、RemoveUser()和UpdateUser()。此外，每个API函数可以包括多个不同的参数列表，诸如UpdateUser(id)、UpdateUser(name)、UpdateUser(firstname,lastname)等。这些API函数可以被生成并通过API注册器对其它服务可用。如下面将更详细描述的，应当注意的是，不要求服务通过API注册器来调用这些函数。代替地，这些函数可用于在API注册器中浏览，并且在被选择时，API注册器可以生成在调用服务中实现这些函数的客户端库。

[0072] 该方法可以附加地包括在API注册器中在API函数和服务的对应端点之间创建绑定(705)。基于上述发现过程和步骤701的注册过程，API注册器现在可以在容器平台中服务的端点和由API注册器创建的API函数之间创建动态绑定。在上面发现可用端点和服务时形成的数据结构中，API注册器现在可以为每个端点存储对应的函数或函数集。如上所述，可以在发现过程确定服务何时被更新、移动、替换或添加到容器平台时不断地更新这个绑定。这允许在调用服务中创建的客户端库首先检查API注册器，以核实或接收用于该服务的当前IP地址和端口号。

[0073] 图7B图示了根据一些实施例的用于向API注册器404注册API的步骤的硬件/软件示意图。如上所述，可以在容器平台210中的容器620中实例化并运行API注册器404。即使容

器平台210表示生产/部署环境，API注册器404仍可以由用于开发服务的IDE 102访问。因此，IDE 102可以提供用于将API定义文件702上传到API注册器404的机制。具体而言，IDE 102的用户界面可以包括允许开发人员定义和/或填充用于API定义文件702的字段的窗口或界面。上面描述的这种信息可以包括函数名称、参数列表、数据类型、字段长度、对象类定义、IP地址和端口号、服务名称或其它唯一标识符等。这种信息可以被上传到API注册器404，并以动态绑定方式链接到端点602的特定IP地址和端口号。最后，API注册器404可以生成可以通过API注册器404可用的一个或多个API函数704。

[0074] 在向API注册器404注册服务并生成一个或多个API函数之后，API注册器可以在开发人员设计服务时使这些函数对开发人员可用。图8图示了根据一些实施例的用于浏览和选择向API注册器804注册的API的图形界面802和命令行界面804的示例。在为容器平台编程和开发新服务时，开发人员可以访问图形界面802以浏览和选择可在他们的服务中使用的API函数。这个图形界面802仅仅是示例，并且不意味着限制可以用于浏览和选择API函数的图形界面的类型。

[0075] 在这个实施例中，IDE 102可以召唤(summon)图形界面802以提供向API注册器注册的API的列表。在这个实施例中，基于端点对API进行分类。例如，与服务对应的一个端点可以提供用于存储用户记录的RESTful接口(例如，“UserStorage”)。图形界面802可以显示通过所选择的端点可用的所有API函数(例如，“CreateUser”、“DeleteUser”、“UpdateUser”等)。在服务提供多个端点的情况下，其它实施例可以基于整个服务对函数进行分组。图形界面802可以接收将在调用服务时使用的一个或多个API函数的选择。然后，API注册器可以提供说明如何使用API函数的文档，包括必需的参数和返回值。本领域普通技术人员将理解的是，命令行界面804可以提供与图形界面802类似的信息并且可以接收类似的输入。

[0076] 图8中所示的界面802、804提供了多个技术优点。首先，这些界面802、804提供向API注册器注册的所有API的最新列表。这与容器平台中当前可用的所有服务的列表对应。服务开发人员可以实时地检索和显示这种信息，而不是需要查找文档、联系服务开发人员和/或执行其它低效任务来定位可用服务的列表。此外，随着服务被更新，API定义文件可以以对应的方式被更新。然后，这更新图8中所示的显示，以提供针对每个API函数的最新可用性信息。

[0077] 图9图示了根据一些实施例的用于使用向API注册器注册的服务及其对应函数的方法的流程图。该方法可以包括提供已注册的API的列表(901)。在期望的服务已知的情况下，可以省略这个步骤。但是，一般而言可以使用上面图8中所述的界面来显示服务以进行浏览和导航。该方法还可以包括接收对API函数的选择(901)。API注册器可以从服务的开发人员接收此选择。例如，开发人员可以决定使用上述的CreateUser()函数来更新用户记录的数据库。图10图示了API注册器可以如何通过用于CreateUser()函数的图形界面802来接收选择1002。其它实施例可以通过命令行界面或通过由IDE提供的其它输入方法来接收选择。

[0078] 返回参考图9，一旦接收到对API函数的选择，API注册器就可以为调用服务生成一个或多个客户端库(905)。生成客户端库可以向调用服务提供动态绑定到API函数的服务端点。具体而言，IDE可以在IDE中生成类对象的集合，类对象封装了直接与容器平台中的服务端点进行接口连接所需的功能。在一些实施例中，客户端库可以包括可以被实例化或用于

调用实施与服务进行通信所需的代码的成员函数的对象类。这些客户端库的示例将在下面更详细地描述。

[0079] 该方法可以附加地包括提供测试数据(907)。当向API注册器注册服务时,它无需是完整的。代替地,服务可以向API注册器指示它尚未准备好提供对调用服务的函数响应。在一些实施例中,上传到API注册器的API定义文件可以包括在服务起作用之前应当返回的信息类型的规范。当调用服务调用API函数时,由API注册器生成的客户端库可以将请求路由到API注册器,而不是服务端点。然后,API注册器可以使用伪值、空值或默认值提供响应。可替代地,客户端库内的代码本身可以生成要返回给调用服务的默认数据。

[0080] 应当认识到的是,图9中所示的具体步骤提供了根据本发明各种实施例的使用API注册器的特定方法。根据替代实施例,也可以执行其它步骤顺序。例如,本发明的替代实施例可以以不同的次序执行上面概述的步骤。而且,图9中所示的各个步骤可以包括多个子步骤,这些子步骤可以以适合于各个步骤的各种顺序执行。此外,取决于特定应用,可以添加或移除附加步骤。本领域普通技术人员将认识到许多变化、修改和替代方案。

[0081] 图11图示了根据一些实施例的由API注册器自动为服务生成的客户端库的示例。这个客户端库1102可以与存储用户记录的服务对应。这个客户端库1102以及对应的类和服务仅通过示例的方式提供,并且不意味着限制。如上所述,每个API函数和服务都可以指定应当如何通过上传到API注册器的API定义文件来生成客户端库。因此,以下关于“User(用户)”服务描述的原理可以应用于其它服务。

[0082] 为了表示“User”服务,API注册器可以为User生成类。当调用服务请求API注册器生成客户端库时,调用服务可以指定该调用服务使用的编程语言。例如,如果调用服务是在IDE中用Java编写的,那么API注册器可以以Java编程语言生成类库。可替代地,如果调用服务是用C#编写的,那么API注册器可以以C#编程语言生成类库。User类可以被生成,以具有与可以通过服务端点执行的不同操作对应的成员函数。这些成员函数可以是静态的,使得它们不要求User类的实例化的实例,或者它们可以与实例化的User对象一起使用。

[0083] 在这个示例中,User服务可以使用RESTful接口编辑由该服务存储的各个用户记录。例如,API注册器可以生成CreateUser()函数以实现对User服务的POST调用。类库可以执行的功能之一是解析、过滤和格式化作为参数被提供给API函数的数据,以作为数据分组被直接发送给服务。在这个示例中,CreateUser()函数可以接受为了方便调用服务而被格式化的参数。例如,调用服务可以分别存储用户名字和用户姓氏的字符串。但是,POST命令可能要求名字和姓氏一起的级联的字符串。为了适应用户友好的参数集,客户端库1102可以执行设置操作,该设置操作将作为函数的参数接收的数据格式化为与服务端点兼容的格式。这可以包括生成报头信息、更改某些数据字段的格式、级联数据字段、从其它源请求附加数据、执行计算或数据变换等。这还可以包括将重新格式化的参数打包为诸如JSON、XML等之类的格式。

[0084] 一旦参数被正确地格式化到用于服务端点的包中,则客户端库1102还可以处置对服务的POST调用。当生成客户端库时,可以将服务的IP地址和端口号插入CreateUser()函数中,以在对服务的HTTP请求中使用。注意的是,HTTP请求的细节被封装在CreateUser()函数中。当调用服务的开发人员想要使用通过该服务可用的POST函数而不是自己在库1102中编写代码时,他们可以代替地从API注册器中选择“User”服务。API注册器然后将自动生

成包括User类的客户端库1102。然后，为了使用POST函数，服务开发人员可以简单地使用User.CreateUser(“John”,“Smith”,2112)函数来将用户John Smith添加到服务。

[0085] 图12图示了根据一些实施例的适应服务端点和API函数之间的动态绑定的客户端库1202的实施例。在这个示例中，当API注册器生成客户端库1202时，CreateUser()函数可以包括动态地检索服务的IP地址和端口号的代码1204。当调用服务114在生产/部署环境104(诸如容器平台)中操作时，调用服务114可以使用GetIPPort()函数在运行时向API注册器404发送请求。API注册器404可以访问其内部表，该内部表被一致地更新以维持API函数和服务端点之间的最新绑定。API注册器404然后可以将当前的IP地址和端口号返回给调用服务114。客户端库1202然后可以将IP地址和端口号插入连接到服务的HTTP POST代码中。因为可以在运行时通过容器平台中的任何调用服务访问API注册器404，所以当被调用的服务的IP地址或端口号改变时，无需更新或修补这些服务。代替地，每次当服务被调用时，API注册器404都可以提供最新信息。在一些实施例中，假设服务端点在生产环境中不会频繁改变，则GetIPPort()函数可以仅需要每小时一次、每天一次、每周一次等调用API注册器404，以最小化在容器外为服务114进行的函数调用的次数。

[0086] 图13图示了根据一些实施例的客户端库1302的实施例，该客户端库1302可以整理附加数据以完成用于服务调用的输入数据集。为了简化客户端库1302的使用，客户端库1302可以最小化服务开发人员所需的参数的数量。进行服务调用所需的附加数据可以从其它源检索，并且因此可以从参数列表中被省略。替代地，客户端库1302可以直接从这些其它源检索这些附加参数。例如，创建新用户可以包括为该用户指定用户角色。代替要求服务开发人员提供用户角色作为参数之一，客户端库1302可以代替地包括自动地从某个其它源检索用户的角色的代码1304。在这个示例中，可以从数据库、从容器平台中的另一个服务或从调用服务内存储用户角色的另一个类检索用户角色。在这些情况中的任何一种情况下，代码1304可以自动检索用户角色并将它打包为发送到服务的HTTP POST命令的输入数据的一部分。

[0087] 除了对用于输入到服务的数据进行整理和格式化之外，客户端库1302还可以解析并返回从服务接收的数据以及处置错误状况。在这个示例中，POST命令可以将数据分组返回到Result变量中。服务常常可以返回包括的信息比调用服务所需的信息更多的数据分组。因此，客户端库1302可以解析Result变量中的数据字段，并提取、格式化来自Result变量的数据，以及将来自Result变量的数据打包成为User类更可用和期望的格式。在这个示例中，代码1306可以从Result变量中提取字段，并使用它们来创建从API函数返回的新的User对象。在另一个使用GET命令的示例中，可以在User类中生成各个API函数，这些API函数从来自GET命令的Result变量中提取不同的字段。例如，User类可以提供GetFirstName(id)函数、GetLastName(id)函数、GetRole(id)函数等。这些函数中的每一个可以包括非常相似的代码，同时返回来自Result变量的不同的字段。

[0088] 除了解析结果之外，客户端库1302还可以生成处置与使用服务相关联的错误状况的代码1308。在这个示例中，代码1308可以测试Result变量中的status字段，以确定POST命令是否成功。如果命令成功，那么CreateUser()函数可以返回新的User对象。在Post命令失败的情况下，该函数可以代替地返回空对象和/或重试对服务的调用。

[0089] 图14图示了根据一些实施例的在调用服务时可以处置重试的客户端库1402。就像

图13的示例一样，客户端库1402使用由POST HTTP调用填充的Result变量中的status来确定该调用是否成功。当结果不成功时，客户端库1402可以继续重试，直到调用成功为止。一些实施例可以使用计数器或其它机制来限制重试的次数或在重试之间添加等待时间。

[0090] 如上所述，一些实施例还可以将API特性的集合连同API定义一起上传到API注册器。图15A图示了根据一些实施例的向API注册器提供API特性的方法。该方法可以包括接收API定义的上传(1501)。该方法还可以包括接收API特性的上传(1503)。特性的上传可以是与API定义的上传相同的传输的一部分。在一些实施例中，API特性可以是API定义的一部分。在一些实施例中，API特性可以是一个或多个标志或预定义的数据字段，该一个或多个标志或预定义的数据字段被检查以指示特性应当由API注册器设置。在一些实施例中，API特性不需要符合任何预先结构化的格式，而是可以代替地由使API注册器实现下述特征(诸如认证、加密等)的指令代码表示。可以将API特性与用于每个服务的API定义一起存储。

[0091] 该方法可以附加地包括在服务和API之间创建API绑定(1505)。可以如上面详细描述的那样执行这个操作。此外，该方法可以包括使用API特性来执行与服务相关联的一个或多个操作(1507)。API特性可以在服务的生命周期期间的不同阶段使用。一般而言，这可以被描述为在服务部署期间、在为服务生成客户端库时和/或在调用服务时使用API特性来实现与该特性关联的函数。这些函数中的每一个的示例将在下面更详细地描述。

[0092] 应当认识到的是，图15A中所示的具体步骤提供了根据本发明各种实施例的向API注册器提供API特性的特定方法。根据替代实施例，也可以执行其它步骤顺序。例如，本发明的替代实施例可以以不同的次序执行上面概述的步骤。而且，图15A中所示的各个步骤可以包括多个子步骤，这些子步骤可以以适合于各个步骤的各种顺序执行。此外，取决于特定应用，可以添加或移除附加步骤。本领域普通技术人员将认识到许多变化、修改和替代方案。

[0093] 图15B图示了根据一些实施例的服务可以如何向API注册器提供API特性的硬件/软件示意图。在IDE 102中开发服务时，服务开发人员可以向API注册器404提供API定义文件1502和一个或多个特性1504。因为在运行时在IDE 102和容器平台两者中API注册器404都是可访问的，所以API注册器404可以存储特性1504并使用它们来影响在开发和运行时场景期间如何部署、调用和/或使用服务来生成客户端库。

[0094] 图16图示了根据一些实施例的硬件/软件示意图，其中API注册器使用特性来部署具有高可用性的服务。除了用于特定服务的API定义文件1505之外，API注册器404还可以接收指示应当将服务部署为非常有弹性或具有高可用性的特性1602。可以接收这个特性1602作为由API注册器404执行以将服务部署为具有高可用性的指令集。这个选项允许开发人员定义对于这个服务“高可用性”意味着什么。例如，特性1602可以包括使API注册器404将服务的多个实例602、604部署到容器平台210的指令。通过执行这些指令，API注册器404不需要自己做出任何决定或确定，而是可以代替地仅执行作为特性1602的一部分提供的部署代码。

[0095] 特性1602也可以作为标志或设置被接收，该标志或设置向API注册器404指示在API注册器404处执行用于部署具有高可用性的服务的现有指令的选项。使用这个选项，API注册器404不需要接收要作为特性1602执行的任何代码。相反，API注册器404可以识别高可用性特性1602并执行在API注册器404中维护的代码以部署服务的多个实例602、604。这允许API注册器404定义对于向API注册器404注册的任何服务的部署而言“高可用性”意味着

什么。

[0096] 因为API注册器404连接到容器平台210的运行时环境，所以API注册器404可以与容器平台210交互以部署确定服务的运行时可用性的实例602、604。注意的是，图16中所示的服务的两个实例602、604仅作为示例提供，并且不意味着限制。高可用性服务可以包括部署到容器平台的服务的多于两个冗余实例。

[0097] 一些实施例可以在API注册器404中包括可以默认执行的代码。如果特性602仅包括期望高可用性的简单指示，那么API注册器404可以执行其自己的代码。如果特性602包括用于部署服务的部署代码，那么API注册器404可以代替地执行特性1602的代码。在一些情况下，特性1602可以仅仅包括部署具有高可用性的服务所需的代码的部分。API注册器404可以执行由特性1602提供的代码的部分，然后使用API注册器404处的代码来执行特性1602未提供的任何代码。这允许开发人员覆写在API注册器404处如何执行特性(诸如高可用性)的现有定义，同时仍允许API注册器404提供用于执行可被已注册的服务使用的特性的统一定义。

[0098] 在一些实施例中，高可用性特性还可以使容器平台210部署负载平衡服务606，该负载平衡服务606将请求分发给该服务的多个实例602、604。负载平衡服务606的端点可以向API注册器404注册，并且对其它服务可用。可替代地或附加地，服务602、604的多个实例中的每一个可以作为服务端点向API注册器404注册。

[0099] 在下面描述的每个示例中，可以应用关于图16讨论的相同原理。例如，以下描述的任何特性都可以附带有可以由API注册器404接收并用于推翻否则将由API注册器404执行的代码的代码。在本公开之前，不存在用于创建用于执行特性的统一默认值、同时允许服务开发人员在需要时推翻那些特性的方法。因此，API注册器404通过允许默认情况下执行在API注册器404处的代码、同时仍允许该代码被从开发人员接收的特性1602推翻来解决技术问题。

[0100] 图17图示了根据一些实施例的通过API注册器强制实施端到端加密的特性的硬件/软件示意图。与API定义文件1505一起，API注册器404可以接收特性1704，该特性指示用于调用服务1708的端到端加密或包括生成用于调用服务1708的端到端加密的代码。在开发期间，服务1708可以包括其自己的使得由服务1708接收的分组被解密并且使由服务1708返回的分组被加密的解密/加密代码1710。在本公开之前，开发人员将需要提供指示服务1708的用户需要提供加密以与服务1708兼容的规范。这个实施例通过允许服务1708指示在调用服务1706中如何生成客户端库来解决技术问题，这确保了与服务1708的加密的兼容性。

[0101] 在一些实施例中，服务1708的开发人员不需要在服务1708中包括加密/解密代码1710。代替地，特性1704可以简单地指示API注册器404强制实施对服务1708的端到端加密。当服务1708被部署到容器平台210时，API注册器404可以使加密/解密代码1710在服务1708被部署时被插入到服务1708中。这允许开发人员基于特性1704在不同的加密方案之间进行选择和/或允许API注册器404选择优选的加密方案作为默认方案。

[0102] 端到端加密不仅要求在部署服务时或在开发期间将加密/解密代码1710插入到服务1708中，而且还要求调用服务1706还包括兼容的加密/解密代码。如上所述，当调用服务1706需要使用服务1708时，API注册器404可以生成一个或多个客户端库1702，该一个或多个客户端库1702以简单且高效的方式完全实现与服务1708交互所需的代码。当生成这个客

户端库1702时，API注册器404可以分析特性1704以确定由服务1708使用的加密方案。然后，基于该特性1704，API注册器404可以使兼容的加密/解密代码被添加到用于调用服务1706的客户端库1702。因此，当调用服务1706向服务1708发送请求时，信息可以在调用服务1706处被加密，并且一旦被服务1708接收，就被解密。类似地，服务1708可以在将响应发送到调用服务1706之前对响应进行加密，然后调用服务1706可以在将该响应传递到客户端库1702外部之前对该响应进行解密。这使得整个加密过程对于调用服务1706的开发人员是完全透明的。特性1704可以确保API注册器404已经在客户端库1702中生成了加密/解密代码以便兼容以及实现端到端加密特性，而不是要求在调用服务1706时实现兼容的加密/解密方案。

[0103] 图18图示了根据一些实施例的用于让API注册器实现服务1808的使用情况日志记录的特性1804。在本公开之前，为了监视和日志记录对服务的请求的频率、源、成功率等，服务本身必须日志记录这种信息。可替代地，容器环境必须监视服务并日志记录其使用情况信息。在服务1808本身处对信息进行日志记录效率非常低，并且减慢了由服务处置的每个请求的吞吐量。类似地，要求容器平台监视和日志记录对特定服务的所有调用的开销也向容器服务的调度和编排带来巨大开销。这个实施例通过将代码直接插入到用于调用服务1808的服务的客户端库中来解决这个技术问题。这允许日志记录和监视服务1808的使用情况，而在存储器使用情况或CPU使用情况方面完全不影响服务1808的性能。

[0104] 除了API定义文件1505之外，API注册器404还可以接收指示使用情况日志记录1804或包括实现使用情况日志记录1804的代码的特性1804。当调用服务1806的开发人员期望向服务1808提交请求时，API注册器404可以自动生成客户端库1802，该客户端库1802包括用于日志记录与服务1808相关的活动的代码。如上所述，这种代码可以基于由API注册器404维护和执行的默认代码来生成，或者可以由利用特性1804接收并由API注册器404执行的代码来生成。

[0105] 用于在客户端库1802中日志记录活动的代码可以包括在服务1808每次被调用时递增的计数器、在服务1808被调用时使得活动被日志记录到日志文件中的函数以及监视和记录发送到服务1808的请求和从服务1808接收的响应的特点的其它函数。取决于特定的实施例，这种代码可以监视与关于服务1808做出的请求相关联的许多不同类型的特点。例如，一些实施例可以日志记录对服务1808进行的调用的总数。一些实施例可以日志记录从服务1808接收的响应的成功率。一些实施例可以日志记录在对服务1808的请求中发送的数据的类型。一些实施例可以日志记录当服务1808被调用时的一天中的时间或其它外部信息。一些实施例可以日志记录可以被用于调试服务1808的去往/来自服务1808的输入和输出分组。一些实施例可以以任何组合日志记录这些特征中的任何特性或所有特性而没有限制。

[0106] 图19图示了根据一些实施例的可以对服务1908强制实施认证协议的特性1904的硬件/软件示意图。一些服务可以要求对用户身份进行认证，并要求用户在对请求做出响应之前被授权以使用该服务。在本公开之前，在服务1908本身处进行认证和授权过程的情况下存在技术问题。对于由服务1908接收的每个调用，这在存储器使用情况和CPU使用情况方面增加了开销，并且增加了响应时服务的延迟。这进而降低了吞吐量，并限制了在任何给定时间间隔期间服务1908可以处理的请求数量。这些实施例通过将认证/授权代码移到由API注册器1404自动生成的客户端库1902来解决这个技术问题。

[0107] 当调用服务1906想要使用服务1908时，API注册器404可以生成包括用于执行授权

和/或认证的代码的客户端库1902。在一些实施例中，这可以包括联系具体地核实用户身份和/或确定用户是否被授权使用服务1908的外部认证/授权服务1920。外部认证/授权服务1920可以包括访问管理器、轻量级目录访问协议(LDAP)管理器、访问控制列表(ACL)、网络认证协议管理器等。然后，当认证/授权过程成功时，客户端库1902中的代码可以将调用发送到服务1908。

[0108] 通过将认证/授权强制实施卸载到API注册器404和客户端库1902，可以从服务1908中完全消除这种代码。因为与外部认证/授权服务1920交互常常附带显著的延迟，所以可以从服务1908中移除这种延迟以增加吞吐量。此外，不是将认证/授权强制实施硬编码到服务1908中，而是服务1908的开发人员可以代替地使用发送到API注册器404的特性1904简单地选择预定义的认证/授权方案。API注册器404可以维护带有用于客户端库1902的附带实现代码的授权/认证的预定义列表。这也防止调用服务1906向服务1908发送不能被授权和/或认证的请求。代替地，如果认证和/或授权例程不成功，那么可以在客户端库1902处终止该调用。这确保服务1908仅接收经过认证和/或授权的请求。

[0109] API注册器404提供的另一个技术改进是升级由特性1904提供的任何功能而无需改变任何已注册的服务的任何代码的能力。例如，因为认证/授权代码已经被卸载到由API注册器1404生成的客户端库1902，所以客户端库1902可以被更新以改变认证/授权方案。调用服务1906或服务1908中的代码均不需要修改。因为仅在单个位置改变代码，所以这大大降低了代码集成错误的概率，否则这些代码集成错误将伴随着发送到每个单独服务的分布式补丁。

[0110] 图20图示了根据一些实施例的用于启用服务的运行时实例化的特性2004的硬件/软件示意图。一些服务可能很少被使用或仅在预定义的时间间隔内被使用。因此，将服务部署到容器平台并不一定总会导致实际上实例化立即可用的容器中的服务实例。与虚拟机相比，可以非常快速地实例化和激活容器。因此，服务开发人员可能期望仅在调用服务时才实例化该服务。服务开发人员可能还期望仅在预定义的时间间隔内实例化该服务。类似地，服务开发人员可以指定应当在预定义的不活动时间间隔之后删除服务实例。

[0111] 除了接收API定义文件1505之外，API注册器404还可以接收指定运行时实例化或其它实例化参数的特性2004。例如，特性可以包括在部署之后对应当实例化服务2008的一个或多个时间间隔的指定。在另一个示例中，特性可以包括仅应当按需实例化服务2008的指示。在另一个示例中，特性可以指定超时间隔，在该超时间隔之后，应当从容器平台删除实例化服务2008。

[0112] 当调用服务2006想要使用服务2008时，API注册器404可以在客户端库2002中生成处置服务2008的运行时实例的代码。例如，客户端库2002中的CreateInstance()函数调用可以创建对API注册器404的调用。API注册器然后可以与容器平台210交互以确定服务2008的操作实例是否可用。如果不可用，那么API注册器404可以指示容器平台210在容器平台2010中的容器中实例化服务2008的实例。容器平台210然后可以将端点(例如，IP地址和端口号)返回给API注册器404。然后，API注册器404可以在该端点和在客户端库2002中创建的API函数调用之间创建绑定。API注册器404然后可以将端点返回到客户端库2002，该客户端库2002可以被用于在调用服务2006和新实例化的服务2008之间创建直接连接。

[0113] 对于仅应当在预定义时间间隔期间被实例化的服务，API注册器404可以为某些服

务建立实例化和删除时间的表。基于这些存储的实例化/删除时间，API注册器404可以指示容器平台210实例化或删除服务2008的实例。API注册器404还可以指定在这些预定义间隔期间应当被实例化的多个实例。例如，从5:00PM到10:00PM，特性2004可以指定服务2008的至少10个实例在容器平台210上处于活动状态。当这个时间间隔发生时，API注册器404可以指示容器平台210创建附加实例。

[0114] 图21图示了根据一些实施例的实现用于服务2108的速率限制功能的特性2104的硬件/软件示意图。一些服务可能需要限制接收请求的速率。其它服务可能需要限制来自某些发件人或服务的类型的请求。在本公开之前，这个功能必须由服务本身通过确定每个请求的源、将源与白名单/黑名单进行比较、并调节它服务于这些请求的速率执行。与上述大多数示例一样，将这个开销放在服务本身中会增加该服务使用的存储器和CPU能力的量，并限制该服务的吞吐量。这些实施例通过在由API注册器生成的客户端库中自动生成速率限制代码来解决这个技术问题。这允许服务借助于特性2104来指定速率限制，而无需服务2108利用其所有相关联的开销来实现该功能。

[0115] 当调用服务2106想要向服务2108发送请求时，API注册器404可以自动生成包括速率限制代码的客户端库2102。当生成客户端库2102时，API注册器404可以确定特定服务2106是否应当受到速率限制。如果不是，那么可以照常生成客户端库2102。如果API注册器404确定调用服务2106应当受到速率限制(例如，通过与白名单/黑名单进行比较)，那么API注册器404可以在客户端库2102中插入添加延迟、添加计数器和/或以其它方式实现速率限制功能的代码，以确保由调用服务2106根据预定义速率在任何给定时间间隔内进行预定义的最大数量的请求。这种代码还可以实现速率限制功能将处于活动状态的时间窗口。这允许服务2108在高业务间隔期间自动实行速率限制。

[0116] 如上所述，系统可以包括IDE 102和生产/部署环境104。生产/部署环境104可以包括云计算平台202。应用和服务可以在IDE 102中开发，并被部署到云计算平台202。通常，IDE 102将包括调试器110。调试器110是专门设计用于测试错误并在服务或应用被开发时定位其中的错误(bug)的软件程序。调试器可以使用指令集模拟器，或者可以直接在底层硬件/处理器上运行程序，以实现对程序指令执行的高级控制。最重要的是，调试器允许开发人员根据由断点表示的特定条件停止或中止程序执行。在程序执行被停止时，开发人员可以检查变量值、程序执行流程和/或应用或服务的状态的其它特点。

[0117] 虽然调试器110在IDE 102中的开发过程中特别有用，但它通常不随服务一起部署到云计算平台202中。部署服务的调试版本将引入技术问题，这些技术问题将阻止云计算平台202的高效使用。例如，在本文中称为“调试版本(build)”的与调试器兼容的服务的版本通常包括与调试器110的兼容性所需的、但对于常规执行而言并非必需的不必要的开销和附加指令。这些附加指令/开销使得服务的调试版本花费更长的时间运行，这增加延迟、降低吞吐量，并使用云计算平台202的更多处理资源。此外，将需要与调试器110软件一起部署调试版本。为服务的每个实例部署调试器110的版本将为服务的每个实例添加大量的附加存储器使用，这进而可能减少云计算平台202的可用存储器资源。

[0118] 因为调试器110没有部署到生产/部署环境104中，所以对于在部署之后发现的错误进行故障排除特别困难。在本公开之前，将使用伪数据或预配置的输入在IDE中运行具有调试器110的服务的调试版本，以尝试并模拟服务在生产/部署环境104中接收到的请求业

务的真实流。但是，这会阻止服务使用实时的实况数据，这些数据对于重新创建错误以及确定针对这些错误的修复/补丁是否在不干扰服务的预期操作的情况下适当工作而言可以是最优的。

[0119] 本文所述的实施例对云计算平台202进行了改变，使得可以在生产/部署环境104中与调试器一起部署服务的调试版本，以从实际请求业务中接收实时的实况数据。在一些实施例中，调试客户端可以发送特定的调试请求，该调试请求通过负载平衡器被路由并且由服务网格指引到服务的调试实例。这允许来自其它客户端的请求通过相同的负载平衡器和服务网格，同时继续直接被路由到服务的正常实例。在一些实施例中，服务网格可以从任何客户端接收请求，并克隆要发送到服务的调试实例的请求。这些实施例允许服务的调试实例与服务的正常实例一起在生产/部署环境104的云计算平台202中操作。这些实施例还通过将实时的实况请求路由到服务的调试实例来解决上述技术问题，而不中断其它服务实例的正常操作。

[0120] 图22图示了根据一些实施例的用于接收服务请求的云计算平台202的一部分的框图。如上所述，一些实施例可以包括可以控制如何在云计算平台202内部署和操作服务的API注册器404。API注册器404可以通信耦合到服务网格2210。服务网格2210是用于微服务或面向服务的环境的可配置基础设施层，其允许服务实例之间的通信是灵活、可靠和快速的。在一些实施例中，服务网格2210可以提供服务发现、负载平衡、加密、认证/授权以及其它能力。在云计算平台202中，随着更多的容器被添加到基础设施，服务网格2210将来自客户端的请求路由到每个服务的特定实例。注意的是，并非在所有实施例中都要求API注册器404。在一些实施例中，下面描述的功能还可以由容器编排层的调度器或其它部件执行。

[0121] 在一些实施例中，服务网格2210可以提供其自己的负载平衡操作。为了清楚起见，图22专门图示了负载平衡器2208，负载平衡器2208可以是服务网格2210的一部分或可以是单独的部件。负载平衡器2208可以被用于将来自多个客户端2202、2204、2206的请求路由到不同的服务实例。不同的服务实例可以驻留在不同的操作环境、不同的容器节点和/或不同的硬件设备上。此外，负载平衡器2212可以与特定服务相关联。单个服务可以具有可用于服务请求的多个服务实例2214、2216、2218。负载平衡器2212可以接收来自服务网格的请求，并将请求业务路由到特定服务实例2214、2016、2218，以使得服务请求均匀地分布在服务实例2214、2016、2218之间。

[0122] 在图22的示例中，服务实例2214、2016、2218中的每一个已经作为服务的生产版本部署到云计算平台202。这些在本文中可以被称为“生产服务”或“正常服务”，并且它们可以与服务的调试实例区分开，因为用于服务实例2214、2016、2218的容器/舱不包括调试器或服务的调试版本。代替地，这些“正常”服务实例2214、2016、2218已经以简化格式进行编译和部署，这允许它们快速且高效地服务于请求，而无需与调试操作相关联的开销。

[0123] 图23图示了根据一些实施例的封装在舱2300中的服务的调试版本。舱2300类似于图3中所示的舱。例如，整个服务可通过端点318可用，并且服务是使用多个微服务或服务320、322、324、326构建的。这些服务可以被打包在多个容器310、312中，并且可以利用公共资源308，诸如存储卷、查找表、数据库或其它资源。

[0124] 但是，与图3中的舱304相比，封装在舱2300中的服务的这个调试版本包括附加的调试实用程序。例如，舱2300包括附加的容器2302，该附加的容器2302包括调试守护进程

2304。一般而言，守护进程是一种计算机程序，它作为后台进程运行，而不是在交互用户的直接控制下运行。在这个特定的容器环境中，调试守护进程2304包括在服务本身的控制之外运行并且被配置为与服务实例对接的后台进程。除了调试守护进程之外，容器2302还可以包括调试登录服务2308，调试登录服务2308允许管理员登录到舱2300并在它操作的同时执行调试操作。容器2302还可以包括附加的调试实用程序2006，诸如在服务接收实时的实况数据请求时跟踪存储器使用情况、处理器使用情况、带宽使用情况和其它计算特点的实用程序。

[0125] 除了容器2302之外，舱2300还可以包括服务本身的特定调试版本。例如，可以使用调试配置来构建或编译服务320、322、324、326中的一个或多个，以使服务320、322、324、326的源代码包括用于调试器进程2310与服务的调试版本一起操作的接口。服务320、322、324、326中的一个或多个将因此能够基于与调试器2310的交互来提供变量值并暂停执行。服务本身的操作可以被认为是来自图22的服务的其它正常实例2214、2216、2218的克隆。因此，图23的舱2300中所示的服务的调试版本可以被用于可靠地调试来自图22的服务的正常实例2214、2216、2218。

[0126] 图24图示了根据一些实施例的用于服务的调试版本的替代舱2400。舱2400类似于图3中的舱306。具体而言，舱2400包括具有通过端点316提供的服务328的单个容器314。但是，容器314也可以加载有提供整体服务的调试版本所需的调试实用程序。具体而言，服务328可以使用调试配置来编译，并被加载到容器314中。此外，调试守护进程2304、调试登录2308、调试器2310和/或附加调试实用程序2306也可以以任何组合且没有限制地包括在容器314中。将理解的是，这些调试实用程序中没有一个是特定实施例要求的，因此一些实施例可以省略上述调试实用程序中的任何一个。

[0127] 在下面的描述中，可以在操作期间在生产/部署环境104中实例化图24中的上面在图23中描述的服务的调试实例。例如，在正常服务实例2214、2016、2218已经被实例化并且在云计算平台202中操作之后，该服务的调试实例也可以被实例化并加载到云计算平台202中。服务的调试实例可以以类似于正常服务实例2214、2216、2218的方式接收常规和/或专用请求业务。这允许在生产环境中实时地调试真实事务，而不中断正常服务实例2214、2216、2218的常规操作。

[0128] 图25图示了根据一些实施例的用于实例化服务的调试实例2502的系统的框图。云计算平台202可以从常规客户端2204、2206接收正常请求。这些正常请求可以通过负载平衡器2208和服务网格2210被路由到特定服务。该服务可以具有多个实例2214、2216、2218，并且用于该服务的负载平衡器2212可以路由请求，使得每个服务实例处的负载被平衡。

[0129] 除了从常规客户端2204、2206接收正常请求之外，一些实施例还可以从调试客户端2504接收调试请求2506。调试客户端2504可以与常规客户端2204、2206完全相同，因此可以包括台式计算机、便携式计算机、工作站、服务器、移动计算设备、智能电话、PDA、平板计算机、膝上型计算机、智能手表等。但是，调试客户端2504可以被具体地识别为向云计算平台202提供调试消息的客户端设备。调试请求2506可以与从常规客户端2204、2206发送的正常请求完全相同。但是，调试请求2506可以由调试客户端2504专门提供，以转到服务的调试实例，而不是服务的正常实例2214、2216、2218。

[0130] 当服务网格2210接收到调试请求2506时，它可以确定服务的调试实例当前是否在

容器平台210中运行。如果当前没有在容器平台210上操作的服务的调试实例，那么服务网格2210可以向API注册器404发送消息以进行生成，可替代地，服务网格2210可以向用于云计算平台202的调度器发起消息，以在不使用API注册器404的实施例中生成服务的调试实例。

[0131] 如上所述，可以通过为服务提供API定义文件1505来完成向API注册器404注册服务。API定义文件1505还可以附带有一个或多个特性，该一个或多个特性表征应如何部署服务和/或应当如何生成用于调用服务的客户端库。在这种情况下，一个或多个特性可以包括动态调试特性2510，该动态调试特性2510向API注册器404指定当服务网格2210辨别或识别出调试请求2506时应当按需生成服务的调试实例。可以部署服务的正常实例2214、2216、2218，而不受动态调试特性2510的影响。在接收调试请求2506之前，不需要在容器平台210上实例化或操作服务的调试实例。

[0132] 在部署服务的正常实例2214、2216、2218之后，并且响应于接收到调试请求2506，API注册器404可以使服务的调试实例2502被部署在云计算平台202中。调试实例2502可以是如图24中所示的单容器实例，或者可以是如图23中所示的多容器实例。在任一实施例中，调试实例都可以包括以任何组合且不加限制的服务的调试版本、调试器、调试登录、调试守护进程和/或任何其它调试实用程序。

[0133] 如同部署到云计算平台202的任何服务一样，服务的调试实例2502可以向API注册器404注册。这可以包括向API注册器404提供端点(例如，IP地址和端口号)。然后，API注册器404可以提供API函数的集合，如上所述，API函数可以被用于调用服务的调试实例2502。可替代地或附加地，由于API注册器404已经部署了服务的调试实例2502，因此服务的调试实例2502不需要向API注册器404提供其自己的API定义文件或特性。代替地，API注册器404可以向服务网格2210提供端点。然后，服务网格2210可以使用端点将调试消息2506路由到服务的调试实例2502。这个端点不需要被发布，因为仅服务网格2210需要向调试实例发送任何消息。但是，一些实施例可以使这个端点可用于在运行时期间访问调试信息的其它服务。

[0134] 图26图示了根据一些实施例的将调试请求路由到服务的调试实例2502的容器平台210的框图。当服务网格2210接收到调试请求2506时，服务网格2210可以识别调试请求2506。可以以多种不同方式将调试请求2506与来自常规客户端2204、2206的正常请求区分开。例如，服务网格2210可以借助于发送者(即，调试客户端2504)来识别调试请求2506。当服务网格2210识别出调试客户端2504的地址时，服务网格2210可以将该请求指定为调试请求2506。这允许服务网格2210凭借调试请求的源来识别调试请求。

[0135] 在一些实施例中，服务网格2210可以使用嵌入在调试请求2506内的任何信息来识别调试请求2506。例如，调试请求2506可以在调试请求2506内具有将其识别为调试请求的唯一的报头信息或一个或多个标志设置。在另一个示例中，调试请求2506可以在请求内包括有效载荷，该有效载荷包括仅由调试请求使用的预定值。

[0136] 当服务网格识别出调试请求2506时，它可以改变调试请求的正常路由。正常请求可以被发送到负载平衡器2212和/或服务的正常实例2214、2216、2218，而调试请求2506可以专门被路由到服务的调试实例2502。例如，可以首先使用来自图25的调试请求2506来触发服务的调试实例2502的实例化。当服务网格2210从API注册器404接收到服务的调试实例

2502可操作的通知时，服务网格2210然后可以将调试请求2506路由到服务的调试实例2502。当从调试客户端2504或其它指定为调试客户端的客户端接收到将来的调试请求时，服务网格2210还可以将这些请求路由到服务的调试实例2502。因为服务的调试实例2502已经在操作，所以可以路由请求而无需来自API注册器404的额外参与。

[0137] 在一些实施例中，服务的调试实例2502可以保持实例化达预定的时间限制。例如，服务的调试实例2502可以包括默认时间限制(例如，24小时)，之后，API注册器404和/或调度器可以删除服务的调试实例2502。在一些实施例中，动态调试特性2510还可以指定专门为该服务定制的时间间隔。例如，服务开发人员可以确定一周是特定服务的更合适的调试间隔，然后可以将七天的间隔作为动态调试特性2510的一部分被提供，以覆盖否则将由API注册器404应用的任何默认时间间隔。在一些实施例中，时间限制可以是绝对的，使得它从实例化开始。在一些实施例中，时间限制可以是相对的，使得每当服务的调试实例2502接收到新的调试请求2506时，就将它重置。

[0138] 图27图示了根据一些实施例的克隆请求的云计算平台202的框图。与上述实施例相反，这些实施例不需要特殊的请求源来向服务的调试实例提供调试请求。代替地，服务网格210可以从常规客户端2202、2204、2206接收正常请求。代替将特殊调试消息与实况数据一起实时地路由到服务的调试实例，这些实施例可以使用实况数据本身并将请求的克隆的副本路由到调试服务，而不会中断服务的正常实例2214、2216、2218的正常操作。

[0139] 如上所述，服务本身可以通过上传API定义文件1505向API注册器404注册。API注册器404可以与API定义文件1505一起接收克隆调试特性2710的上传。克隆调试特性2710可以向API注册器404指示应当在云计算平台202中实例化服务的调试实例2502。当部署服务的正常实例2214、2216、2218时，API注册器404和/或调度器可以部署服务的调试实例2502。可替代地或附加地，可以响应于从客户端接收到被发送到服务的请求而部署服务的调试实例2502。对于使用API注册器404的实施例，API注册器可以将服务的调试实例2502的端点提供给服务网格2210。

[0140] API注册器404和/或调度器还可以向服务网格2210提供指示发送到服务的请求应当被克隆并且还应当被发送到服务的调试实例2502的指示2702。指示2702可以指示所有请求应当被克隆并被发送到服务的调试实例2502。在一些实施例中，指示2702可以指示应当为服务的调试实例2502克隆一定百分比的请求(例如，每隔一个请求、每第三个请求等等)。

[0141] 图28图示了根据一些实施例的克隆的请求被转发到服务的调试实例2502的框图。当从常规客户端2202发送正常请求2806时，正常请求2806可以经过负载均衡器2208并由服务网格2210识别。服务网格2210可以将正常请求2806识别为被寻址到服务。然后，基于指示2702和/或这种识别，服务网格2010然后可以生成克隆的请求2808。在一些实施例中，克隆的请求2808可以是正常请求2006的完整副本。在一些实施例中，克隆的请求2808可以包括由服务网格2210插入的在调试过程中可能有用的附加信息(例如，时间戳或在调试场景中可能有用的其它诊断信息)。

[0142] 然后，可以将克隆的请求2808转发到服务的调试实例2502。这可以与将正常请求2806转发到服务的正常实例2214、2216、2218并行地完成。因此，这个克隆过程可以在不影响服务的正常实例2214、2016、2218的吞吐量的情况下操作。这还允许服务的调试实例2502将实时的实况数据用于调试过程。因此，在服务的正常实例2214、2216、2218中发生的任何

错误也将被服务的调试实例2502的输出捕获。这提供了独特的技术优势，其允许使用实况调试数据在问题发生时实时地隔离它们，而不会产生仅由服务的调试实例处理正常请求2806通常所附带的开销。

[0143] 图29图示了用于为容器环境中的容器化的服务提供运行时调试的方法的流程图。该方法可以包括接收对在容器环境处的服务的请求(2902)。容器环境可以包括服务网格和封装在多个容器中的多个服务。服务也可以封装在容器环境中的第一一个或多个容器中。容器环境可以包括上述的容器平台210。请求可以是来自常规客户端设备的调试请求或正常请求。可以将第一一个或多个容器组织到容器舱中，该容器舱可以包括形成服务的一个或多个微服务。容器环境可以包括带有容器调度器的编排的容器平台。容器环境还可以包括API注册器。

[0144] 该方法还可以包括确定应当将请求路由到服务的调试实例(2904)。可以基于许多不同的因素(包括请求的源、请求中的报头、请求的有效载荷中的值、请求中的标志、接收请求的定时和/或请求的任何其它特点)做出这个确定。如果请求不是调试请求，那么该方法可以包括将请求路由到服务的正常实例(2908)。

[0145] 如果该请求被识别为调试请求，那么该方法还可以包括确定服务的调试实例是否可用(2906)。如果调试实例可用，那么该方法可以包括将请求路由到服务的调试实例(2912)。如果调试实例不可用，那么该方法还可以包括实例化服务的调试实例(2910)，以及将请求路由到新实例化的服务的调试实例(2912)。

[0146] 应当认识到的是，根据本发明的各种实施例，图29中所示的特定步骤提供了在容器环境中实现实况调试的特定方法。根据替代实施例，也可以执行其它步骤顺序。例如，本发明的替代实施例可以以不同的次序执行上面概述的步骤。而且，图29中所示的各个步骤可以包括多个子步骤，这些子步骤可以按照适合于各个步骤的各种顺序来执行。此外，取决于特定应用，可以添加或移除附加步骤。本领域普通技术人员将认识到许多变化、修改和替代。

[0147] 本文描述的每个方法可以由专用计算机系统来实现。这些方法的每个步骤可以由计算机系统自动执行，和/或可以被提供有涉及用户的输入/输出。例如，用户可以为方法中的每个步骤提供输入，并且这些输入中的每一个可以响应于请求这种输入的具体输出，其中输出由计算机系统生成。每个输入可以响应于对应的请求输出而被接收。此外，输入可以从用户接收、作为数据流从另一个计算机系统接收、从存储器位置检索、通过网络检索、从web服务请求等。同样，输出可以提供给用户、作为数据流提供给另一个计算机系统、保存在存储器位置中、通过网络发送、被提供给web服务等。简而言之，本文描述的方法的每个步骤可以由计算机系统执行，并且可以涉及到计算机系统和来自计算机系统的任何数量的输入、输出和/或请求，这些输入、输出和/或请求可以涉及或可以不涉及用户。可以被称为不涉及用户的那些步骤是由计算机系统自动执行的而无需人为干预。因此，根据本公开将理解的是，可以更改本文描述的每个方法的每个步骤以包括到用户和来自用户的输入和输出，或者可以由计算机系统自动完成而无需人为干预，其中任何确定由处理器产生。此外，本文描述的每个方法的一些实施例可以被实现为存储在有形的、非瞬态存储介质上的指令集合，以形成有形的软件产品。

[0148] 图30描绘了可以与上述任何实施例进行交互的分布式系统3000的简化图。在所示

实施例中，分布式系统3000包括一个或多个客户端计算设备3002、3004、3006和3008，该一个或多个客户端计算设备可以被配置为通过一个或多个网络3010执行和操作客户端应用，诸如web浏览器、专有客户端(例如，Oracle Forms)等。服务器3012可以经由网络3010与远程客户端计算设备3002、3004、3006和3008通信地耦合。

[0149] 在各种实施例中，服务器3012可以适于运行由系统的一个或多个部件提供的一个或多个服务或软件应用。在一些实施例中，这些服务可以作为基于web的服务或云服务被提供，或者在软件即服务(SaaS)模型下被提供给客户端计算设备3002、3004、3006和/或3008的用户。操作客户端计算设备3002、3004、3006和/或3008的用户又可以利用一个或多个客户端应用来与服务器3012交互以利用由这些部件提供的服务。

[0150] 在图中绘出的配置中，系统3000的软件部件3018、3020和3022被示出为在服务器3012上实现。在其它实施例中，系统3000的一个或多个部件和/或由这些部件提供的服务也可以由客户端计算设备3002、3004、3006和/或3008中的一个或多个来实现。然后，操作客户端计算设备的用户可以利用一个或多个客户端应用来使用由这些部件提供的服务。这些部件可以用硬件、固件、软件或其组合来实现。应该认识到的是，各种不同的系统配置是可能的，这些系统配置可能与分布式系统3000不同。图中所示的实施例因此是用于实现实施例系统的分布式系统的一个示例，而不旨在进行限制。

[0151] 客户端计算设备3002、3004、3006和/或3008可以是便携式手持设备(例如，**iPhone®**、蜂窝电话、**iPad®**、计算平板电脑、个人数字助理(PDA))或可穿戴设备(例如，Google **Glass®**头戴式显示器)，运行诸如Microsoft Windows **Mobile®**和/或各种移动操作系统(诸如iOS、Windows Phone、Android、BlackBerry 10、Palm OS等)之类的软件，并且启用互联网、电子邮件、短消息服务(SMS)、**Blackberry®**或其它通信协议。客户端计算设备可以是通用个人计算机，作为示例，包括运行各种版本的Microsoft **Windows®**、Apple **Macintosh®**和/或Linux操作系统的个人计算机和/或膝上型计算机。客户端计算设备可以是运行各种可商业获得的**UNIX®**或类UNIX操作系统(包括但不限于各种GNU/Linux操作系统，诸如例如Google Chrome OS)中的任何操作系统的工作站计算机。替代地或附加地，客户端计算设备3002、3004、3006和3008可以是能够通过(一个或多个)网络3010通信的任何其它电子设备，诸如瘦客户端计算机、启用互联网的游戏系统(例如，具有或不具有**Kinect®**姿势输入设备的微软Xbox游戏控制台)和/或个人消息传送设备。

[0152] 虽然示例性分布式系统3000被示出为具有四个客户端计算设备，但是可以支持任何数量的客户端计算设备。其它设备(诸如具有传感器的设备等)可以与服务器3012交互。

[0153] 分布式系统3000中的(一个或多个)网络3010可以是本领域技术人员熟悉的、可以利用各种可商业获得的协议中的任何协议支持数据通信的任何类型的网络，其中协议包括但不限于TCP/IP(传输控制协议/互联网协议)、SNA(系统网络体系架构)、IPX(互联网分组交换)、AppleTalk，等等。仅仅作为示例，(一个或多个)网络3010可以是局域网(LAN)，诸如基于以太网、令牌环等的LAN。(一个或多个)网络3010可以是广域网和互联网。它可以包括虚拟网络，包括但不限于虚拟专用网(VPN)、内联网、外联网、公共交换电话网(PSTN)、红外网络、无线网络(例如，依据电子电气学会(IEEE)802.11协议套件、**Bluetooth®**和/或任

何其它无线协议当中的任意一种操作的网络)；和/或这些和/或其它网络的任意组合。

[0154] 服务器3012可以包括一个或多个通用计算机、专用服务器计算机(作为示例，包括PC(个人计算机)服务器、UNIX®服务器、中型服务器、大型计算机、机架安装的服务器等)、服务器场、服务器集群或任何其它适当的布置和/或组合。在各种实施例中，服务器3012可以适于运行在前述公开中所描述的一个或多个服务或软件应用。例如，服务器3012可以与用于执行根据本公开的实施例的以上描述的处理的服务器对应。

[0155] 服务器3012可以运行包括以上讨论的操作系统当中任意一种的操作系统，以及任何可商业获得的服务器操作系统。服务器3012还可以运行各种附加的服务器应用和/或中间层应用中的任何服务器应用和/或中间层应用，包括HTTP(超文本传输协议)服务器、FTP(文件传输协议)服务器、CGI(公共网关接口)服务器、JAVA®服务器、数据库服务器，等等。示例性数据库服务器包括但不限于从Oracle、Microsoft、Sybase、IBM(国际商业机器)等可商业获得的那些数据库服务器。

[0156] 在一些实施方式中，服务器3012可以包括一个或多个应用，以分析和整合从客户端计算设备3002、3004、3006和3008的用户接收到的数据馈送和/或事件更新。作为示例，数据馈送和/或事件更新可以包括但不限于Twitter®馈送、Facebook®更新或者从一个或多个第三方信息源接收到的实时更新和连续数据流，其可以包括与传感器数据应用、金融报价机、网络性能测量工具(例如，网络监视和业务管理应用)、点击流分析工具、汽车交通监视等相关的实时事件。服务器3012还可以包括一个或多个应用，以经由客户端计算设备3002、3004、3006和3008的一个或多个显示设备显示数据馈送和/或实时事件。

[0157] 分布式系统3000还可以包括一个或多个数据库3014和3016。数据库3014和3016可以驻留在各种位置中。作为示例，数据库3014和3016中的一个或多个可以驻留在服务器3012本地的(和/或驻留在服务器3012中的)非瞬态存储介质上。替代地，数据库3014和3016可以远离服务器3012，并且经由基于网络的连接或专用的连接与服务器3012通信。在一组实施例中，数据库3014和3016可以驻留在存储区域网络(SAN)中。类似地，用于执行服务器3012所具有的功能的任何必要的文件都可以适当地本地存储在服务器3012上和/或远程存储。在一组实施例中，数据库3014和3016可以包括适于响应于SQL格式的命令而存储、更新和检索数据的关系数据库，诸如由Oracle提供的数据库。

[0158] 图31是根据本公开的实施例的系统环境3100的一个或多个部件的简化框图，通过该系统环境3100，由实施例系统的一个或多个部件提供的服务可以作为云服务提供。在所示实施例中，系统环境3100包括可以由用户使用以与提供云服务的云基础设施系统3102交互的一个或多个客户端计算设备3104、3106和3108。客户端计算设备可以被配置为操作客户端应用，诸如web浏览器、专有客户端应用(例如，Oracle Forms)或某种其它应用，客户端应用可以由客户端计算设备的用户用来与云基础设施系统3102交互以使用由云基础设施系统3102提供的服务。

[0159] 应该认识到的是，图中描绘的云基础设施系统3102可以具有除了所描绘的部件之外的其它部件。另外，图中所示的实施例仅是可以结合本发明的实施例的云基础设施系统的一个示例。在一些其它实施例中，云基础设施系统3102可以具有比图中所示更多或更少的部件、可以组合两个或更多个部件、或者可以具有不同的部件配置或布置。

[0160] 客户端计算设备3104、3106和3108可以是与上面针对3002、3004、3006和3008所描述的设备类似的设备。

[0161] 虽然示例性系统环境3100被示出具有三个客户端计算设备，但是任何数量的客户端计算设备可以被支持。诸如具有传感器的设备等之类的其它设备可以与云基础设施系统3102交互。

[0162] (一个或多个)网络3110可以促进客户端3104、3106和3108与云基础设施系统3102之间的数据通信和交换。每个网络可以是本领域技术人员所熟悉的可以使用各种商业上可获得的协议(包括上面针对(一个或多个)网络3010所描述的协议)中的任何一种支持数据通信的任何类型的网络。

[0163] 云基础设施系统3102可以包括一个或多个计算机和/或服务器，其可以包括上面针对服务器3012所描述的那些计算机和/或服务器。

[0164] 在某些实施例中，由云基础设施系统提供的服务可以包括按需对云基础设施系统的用户可用的许多服务，诸如在线数据存储和备份解决方案、基于Web的电子邮件服务、被托管的办公套件和文档协作服务、数据库处理、受管理的技术支持服务等。由云基础设施系统提供的服务可以动态缩放以满足云基础设施系统的用户的需要。由云基础设施系统提供的服务的具体实例化在本文中被称为“服务实例”。一般而言，从云服务提供商的系统经由通信网络(诸如互联网)对用户可用的任何服务被称为“云服务”。通常，在公共云环境中，构成云服务提供商的系统的服务器和系统与客户自己的内部部署的服务器和系统不同。例如，云服务提供商的系统可以托管应用，并且用户可以经由诸如互联网之类的通信网络按需订购和使用应用。

[0165] 在一些示例中，计算机网络云基础设施中的服务可以包括对存储装置、被托管的数据库、被托管的Web服务器、软件应用或由云供应商向用户提供的其它服务的受保护的计算机网络访问，或者如本领域中另外已知的那样。例如，服务可以包括通过互联网对云上的远程存储装置进行密码保护的访问。作为另一个示例，服务可以包括基于web服务的被托管的关系数据库和脚本语言中间件引擎，以供联网的开发人员私有使用。作为另一个示例，服务可以包括对在云供应商的网站上托管的电子邮件软件应用的访问。

[0166] 在某些实施例中，云基础设施系统3102可以包括以自助服务、基于订阅、弹性可扩展、可靠、高度可用和安全的方式交付给客户的应用、中间件和数据库服务产品的套件。这种云基础设施系统的示例是由本受让人提供的Oracle公共云。

[0167] 在各种实施例中，云基础设施系统3102可以适于自动供应、管理和跟踪客户对由云基础设施系统3102供给的服务的订阅。云基础设施系统3102可以经由不同的部署模型来提供云服务。例如，可以依据公共云模型提供服务，在公共云模型中云基础设施系统3102被销售云服务的组织拥有(例如，被Oracle拥有)，并且服务对一般公众或不同行业的企业可用。作为另一个示例，可以依据私有云模型来提供服务，在私有云模型中云基础设施系统3102仅针对单个组织操作，并且可以为该组织内的一个或多个实体提供服务。还可以依据社区云模型来提供云服务，在社区云模型中云基础设施系统3102和由云基础设施系统3102提供的服务由相关社区中的若干组织共享。云服务还可以依据混合云模型被提供，该混合云模型是两个或更多个不同模型的组合。

[0168] 在一些实施例中，由云基础设施系统3102提供的服务可以包括在软件即服务

(SaaS)类别、平台即服务(PaaS)类别、基础设施即服务(IaaS)类别或包括混合服务的其它服务类别下提供的一个或多个服务。客户经由订阅订单可以订购由云基础设施系统3102提供的一个或多个服务。云基础设施系统3102然后执行处理以提供客户的订阅订单中的服务。

[0169] 在一些实施例中,由云基础设施系统3102提供的服务可以包括但不限于应用服务、平台服务和基础设施服务。在一些示例中,应用服务可以由云基础设施系统经由SaaS平台提供。SaaS平台可以被配置为提供落入SaaS类别的云服务。例如,SaaS平台可以提供在集成开发和部署平台上构建和交付按需应用套件的能力。SaaS平台可以管理和控制用于提供SaaS服务的底层软件和基础设施。通过利用由SaaS平台提供的服务,客户可以利用在云基础设施系统上执行的应用。客户可以获取应用服务,而无需客户购买单独的许可和支持。可以提供各种不同的SaaS服务。示例包括但不限于为大型组织提供销售业绩管理、企业集成和业务灵活性的解决方案的服务。

[0170] 在一些实施例中,平台服务可以由云基础设施系统经由PaaS平台提供。PaaS平台可以被配置为提供落入PaaS类别的云服务。平台服务的示例可以包括但不限于使组织(诸如Oracle)能够在共享的公共体系架构上整合现有应用以及充分利用平台提供的共享服务来构建新应用的能力的服务。PaaS平台可以管理和控制用于提供PaaS服务的底层软件和基础设施。客户可以获取由云基础架构系统提供的PaaS服务,而无需客户购买单独的许可和支持。平台服务的示例包括但不限于Oracle Java云服务(JCS)、Oracle数据库云服务(DBCS)等。

[0171] 通过利用由PaaS平台提供的服务,客户可以采用由云基础设施系统支持的编程语言和工具,并且还可以控制所部署的服务。在一些实施例中,由云基础设施系统提供的平台服务可以包括数据库云服务、中间件云服务(例如,Oracle融合中间件服务)和Java云服务。在一个实施例中,数据库云服务可以支持共享服务部署模型,该模型使得组织能够汇集数据库资源并且以数据库云的形式向客户供应数据库即服务。在云基础设施系统中,中间件云服务可以为客户提供开发和部署各种业务应用的平台,并且Java云服务可以为客户提供部署Java应用的平台。

[0172] 各种不同的基础设施服务可以由云基础设施系统中的IaaS平台提供。基础设施服务促进底层计算资源(诸如存储装置、网络和其它基础计算资源)的管理和控制,以供客户利用由SaaS平台和PaaS平台提供的服务。

[0173] 在某些实施例中,云基础设施系统3102还可以包括基础设施资源3130,用于向云基础设施系统的客户提供用于提供各种服务的资源。在一个实施例中,基础设施资源3130可以包括预先集成和优化的硬件(诸如服务器、存储装置和联网资源)的组合,以执行由PaaS平台和SaaS平台提供的服务。

[0174] 在一些实施例中,云基础设施系统3102中的资源可以由多个用户共享并且根据需要动态重新分配。此外,可以将资源分配给在不同时区的用户。例如,云基础设施系统3130可以使在第一时区中的第一组用户能够在指定的小时数内利用云基础设施系统的资源,并且然后使相同资源能够被重新分配给位于不同时区的另一组用户,从而使资源的利用率最大化。

[0175] 在某些实施例中,可以提供由云基础设施系统3102的不同部件或模块以及由云基

础设施系统3102提供的服务共享的多个内部共享服务3132。这些内部共享服务可以包括但不限于:安全和身份服务、集成服务、企业储存库服务、企业管理器服务、病毒扫描和白名单服务、高可用性、备份和恢复服务、启用云支持的服务、电子邮件服务、通知服务、文件传输服务等。

[0176] 在某些实施例中,云基础设施系统3102可以提供云基础设施系统中的云服务(例如,SaaS、PaaS和IaaS服务)的综合管理。在一个实施例中,云管理功能可以包括用于供应、管理和跟踪由云基础设施系统3102接收到的客户订阅等的能力。

[0177] 在一个实施例中,如图中所绘出的,云管理功能可以由一个或多个模块提供,诸如订单管理模块3120、订单编排模块3122、订单供应模块3124、订单管理和监视模块3126,以及身份管理模块3128。这些模块可以包括一个或多个计算机和/或服务器或者使用一个或多个计算机和/或服务器来提供,该一个或多个计算机和/或服务器可以是通用计算机、专用服务器计算机、服务器场、服务器集群或任何其它适当的布置和/或组合。

[0178] 在示例性操作3134中,使用客户端设备(诸如客户端设备3104、3106或3108)的客户可以通过请求由云基础设施系统3102提供的一个或多个服务并且对由云基础设施系统3102供应的一个或多个服务下订阅订单来与云基础设施系统3102交互。在某些实施例中,客户可以访问云用户界面(UI)(云UI 3112、云UI 3114和/或云UI3116)并经由这些UI下订阅订单。云基础设施系统3102响应于客户下订单而接收到的订单信息可以包括识别客户以及客户想要订阅的云基础设施系统3102供应的一个或多个服务的信息。

[0179] 在客户下订单之后,经由云UI 3112、3114和/或3116接收订单信息。

[0180] 在操作3136处,订单存储在订单数据库3118中。订单数据库3118可以是由云基础设施系统3118操作和与其它系统元件一起操作的若干数据库之一。

[0181] 在操作3138处,订单信息被转发到订单管理模块3120。在一些情况下,订单管理模块3120可以被配置为执行与订单相关的计费和记账功能,诸如验证订单、以及在验证后预订订单。

[0182] 在操作3140处,将关于订单的信息传送到订单编排模块3122。订单编排模块3122可以利用订单信息为客户下的订单编排服务和资源的供应。在一些情况下,订单编排模块3122可以使用订单供应模块3124的服务来编排资源的供应以支持所订阅的服务。

[0183] 在某些实施例中,订单编排模块3122使得能够管理与每个订单相关联的业务过程并应用业务逻辑来确定订单是否应该继续供应。在操作3142处,在接收到新订阅的订单时,订单编排模块3122向订单供应模块3124发送分配资源并配置履行订阅订单所需的那些资源的请求。订单供应模块3124使得能够为客户订购的服务分配资源。订单供应模块3124提供在由云基础设施系统3100提供的云服务和用于供应用于提供所请求的服务的资源的物理实现层之间的抽象层。因此,订单编排模块3122可以与实现细节(诸如服务和资源是实际上即时供应还是预先供应并仅在请求后才分配/指派)隔离。

[0184] 在操作3144处,一旦供应了服务和资源,就可以通过云基础设施系统3102的订单供应模块3124向客户端设备3104、3106和/或3108上的客户发送所提供的服务的通知。

[0185] 在操作3146处,订单管理和监视模块3126可以管理和跟踪客户的订阅订单。在一些情况下,订单管理和监视模块3126可以被配置为收集订阅订单中的服务的使用统计信息,诸如,所使用的存储量、传输的数据量、用户的数量以及系统运行时间量和系统停机时

间量。

[0186] 在某些实施例中，云基础设施系统3100可以包括身份管理模块3128。身份管理模块3128可以被配置为提供身份服务，诸如云基础设施系统3100中的访问管理和授权服务。在一些实施例中，身份管理模块3128可以控制关于希望利用由云基础设施系统3102提供的服务的客户的信息。这样的信息可以包括认证这些客户的身份的信息以及描述这些客户被授权相对于各种系统资源(例如，文件、目录、应用、通信端口、存储器段等)执行哪些动作的信息。身份管理模块3128还可以包括对关于每个客户的描述性信息以及关于可以如何和由谁来访问和修改该描述性信息的管理。

[0187] 图32示出了其中可以实现本发明的各种实施例的示例性计算机系统3200。系统3200可以用于实现上述任何计算机系统。如图所示，计算机系统3200包括经由总线子系统3202与多个外围子系统通信的处理单元3204。这些外围子系统可以包括处理加速单元3206、I/O子系统3208、存储子系统3218和通信子系统3224。存储子系统3218包括有形计算机可读存储介质3222和系统存储器3210。

[0188] 总线子系统3202提供用于让计算机系统3200的各种部件和子系统按意图彼此通信的机制。虽然总线子系统3202被示意性地示出为单条总线，但是总线子系统的替代实施例可以利用多条总线。总线子系统3202可以是使用各种总线体系架构中的任何总线体系架构的若干种类型的总线结构中的任何一种，包括存储器总线或存储器控制器、外围总线、以及局部总线。例如，这种体系架构可以包括工业标准体系架构(ISA)总线、微通道体系架构(MCA)总线、增强型ISA(EISA)总线、视频电子标准协会(VESA)局部总线和外围部件互连(PCI)总线，其可以被实现为按IEEE P1386.1标准制造的Mezzanine总线。

[0189] 可以被实现为一个或多个集成电路(例如，常规微处理器或微控制器)的处理单元3204控制计算机系统3200的操作。一个或多个处理器可以被包括在处理单元3204中。这些处理器可以包括单核或多核处理器。在某些实施例中，处理单元3204可以被实现为一个或多个独立的处理单元3232和/或3234，其中在每个处理单元中包括单核或多核处理器。在其它实施例中，处理单元3204还可以被实现为通过将两个双核处理器集成到单个芯片中形成的四核处理单元。

[0190] 在各种实施例中，处理单元3204可以响应于程序代码执行各种程序并且可以维护多个并发执行的程序或进程。在任何给定的时间，要被执行的程序代码中的一些或全部代码可以驻留在(一个或多个)处理器3204中和/或存储子系统3218中。通过适当的编程，(一个或多个)处理器3204可以提供上述各种功能。计算机系统3200可以附加地包括处理加速单元3206，其可以包括数字信号处理器(DSP)、专用处理器，等等。

[0191] I/O子系统3208可以包括用户接口输入设备和用户接口输出设备。用户接口输入设备可以包括键盘、诸如鼠标或轨迹球之类的定点设备、结合到显示器中的触摸板或触摸屏、滚动轮、点击轮、拨盘、按钮、开关、键盘、具有语音命令识别系统的音频输入设备、麦克风以及其它类型的输入设备。用户接口输入设备可以包括，例如，运动感测和/或姿势识别设备，诸如的Microsoft **Kinect®**运动传感器，其使得用户能够使用姿势和语音命令通过自然用户接口来控制诸如的Microsoft **Xbox®** 360游戏控制器之类的输入设备并与该输入设备交互。用户接口输入设备也可以包括眼睛姿势识别设备，诸如从用户检测眼睛活动(例如，当拍摄照片和/或做出菜单选择时的“眨眼”)并且将眼睛姿势转换为到输入设备(例

如,Google Glass®)中的输入的Google Glass®眨眼检测器。此外,用户接口输入设备可以包括使用户能够通过语音命令与语音识别系统(例如,Siri®导航器)交互的语音识别感测设备。

[0192] 用户接口输入设备也可以包括但不限于三维(3D)鼠标、操纵杆或指向棒、游戏板和绘图板以及音频/视频设备,诸如扬声器、数码相机、数码摄像机、便携式媒体播放器、网络摄像头、图像扫描仪、指纹扫描仪、条形码阅读器3D扫描仪、3D打印机、激光测距仪和视线跟踪设备。此外,用户接口输入设备可以包括,例如,医学成像输入设备,诸如计算机断层扫描、磁共振成像、正电子发射断层摄影术、医疗超声设备。用户接口输入设备还可以包括,例如,诸如MIDI键盘、数字乐器等之类的音频输入设备。

[0193] 用户接口输出设备可以包括显示子系统、指示灯,或者诸如音频输出设备之类的非可视显示器,等等。显示子系统可以是阴极射线管(CRT)、诸如使用液晶显示器(LCD)或等离子显示器的平板设备、投影设备、触摸屏,等等。一般而言,术语"输出设备"的使用意在包括用于从计算机系统3200向用户或其它计算机输出信息的所有可能类型的设备和机制。例如,用户接口输出设备可以包括,但不限于,可视地传达文本、图形和音频/视频信息的各种显示设备,诸如监视器、打印机、扬声器、耳机、汽车导航系统、绘图仪、语音输出设备以及调制解调器。

[0194] 计算机系统3200可以包括包含软件元件、被示为当前位于系统存储器3210中的存储子系统3218。系统存储器3210可以存储可加载并且可在处理单元3204上执行的程序指令,以及在这些程序的执行期间所产生的数据。

[0195] 取决于计算机系统3200的配置和类型,系统存储器3210可以是易失性的(诸如随机存取存储器(RAM))和/或非易失性的(诸如只读存储器(ROM)、闪存存储器,等等)。RAM通常包含可被处理单元3204立即访问和/或目前正被处理单元3204操作和执行的数据和/或程序模块。在一些实现中,系统存储器3210可以包括多种不同类型的存储器,例如静态随机存取存储器(SRAM)或动态随机存取存储器(DRAM)。在一些实现中,包含有助于诸如在启动期间在计算机系统3200的元件之间传送信息的基本例程的基本输入/输出系统(BIOS)通常可以被存储在ROM中。作为示例,但不是限制,系统存储器3210也示出了可以包括客户端应用、web浏览器、中间层应用、关系数据库管理系统(RDBMS)等的应用程序3212、程序数据3214以及操作系统3216。作为示例,操作系统3216可以包括各种版本的Microsoft Windows®,Apple Macintosh®和/或Linux操作系统、各种可商业获得的UNIX®或类UNIX操作系统(包括但不限于各种GNU/Linux操作系统、Google Chrome® OS等)和/或诸如iOS、Windows® Phone、Android® OS、BlackBerry® 10OS和Palm® OS操作系统之类的移动操作系统。

[0196] 存储子系统3218还可以提供用于存储提供一些实施例的功能的基本编程和数据结构的有形计算机可读存储介质。当被处理器执行时提供上述功能的软件(程序、代码模块、指令)可以被存储在存储子系统3218中。这些软件模块或指令可以被处理单元3204执行。存储子系统3218还可以提供用于存储根据本发明被使用的数据的储存库。

[0197] 存储子系统3200还可以包括可以被进一步连接到计算机可读存储介质3222的计算机可读存储介质读取器3220。通过与系统存储器3210一起并且,可选地与系统存储器

3210相结合，计算机可读存储介质3222可以全面地表示用于临时和/或更持久地包含、存储、发送和检索计算机可读信息的远程、本地、固定和/或可移除存储设备加存储介质。

[0198] 包含代码或代码的部分的计算机可读存储介质3222还可以包括本领域已知或使用的任何适当的介质，包括存储介质和通信介质，诸如但不限于，以用于信息的存储和/或传输的任何方法或技术实现的易失性和非易失性、可移除和不可移除介质。这可以包括有形的计算机可读存储介质，诸如RAM、ROM、电可擦除可编程ROM(EEPROM)、闪存存储器或其它存储器技术、CD-ROM、数字多功能盘(DVD)或其它光学存储装置、磁带盒、磁带、磁盘存储装置或其它磁存储设备、或者其它有形的计算机可读介质。这还可以包括非有形的计算机可读介质，诸如数据信号、数据传输，或者可以被用来发送期望信息并且可以被计算系统3200访问的任何其它介质。

[0199] 作为示例，计算机可读存储介质3222可以包括从不可移除的非易失性磁介质读取或对不可移除的非易失性磁介质写入的硬盘驱动器、从可移除的非易失性磁盘读取或对可移除的非易失性磁盘写入的磁盘驱动器、以及从可移除的非易失性光盘(诸如CD ROM、DVD和Blu-**Ray®**盘或其它光学介质)读取或对可移除的非易失性光盘写入的光盘驱动器。计算机可读存储介质3222可以包括但不限于**Zip®**驱动器、闪存卡、通用串行总线(USB)闪存驱动器、安全数字(SD)卡、DVD盘、数字音频带，等等。计算机可读存储介质3222还可以包括基于非易失性存储器的固态驱动器(SSD)(诸如基于闪存存储器的SSD、企业闪存驱动器、固态ROM等)、基于易失性存储器的SSD(诸如固态RAM、动态RAM、静态RAM、基于DRAM的SSD、磁阻RAM(MRAM)SSD)、以及使用基于DRAM的SSD和基于闪存存储器的SSD的组合的混合SSD。盘驱动器及其关联的计算机可读介质可以为计算机系统3200提供计算机可读指令、数据结构、程序模块及其它数据的非易失性存储。

[0200] 通信子系统3224提供到其它计算机系统和网络的接口。通信子系统3224用作用于从其它系统接收数据和从计算机系统3200向其它系统发送数据的接口。例如，通信子系统3224可以使计算机系统3200能够经由互联网连接到一个或多个设备。在一些实施例中，通信子系统3224可以包括用于(例如，使用蜂窝电话技术、诸如3G、4G或EDGE(用于全球演进的增强型数据速率)之类的先进数据网络技术、WiFi(IEEE 802.11系列标准)、或其它移动通信技术、或其任意组合))访问无线语音和/或数据网络的射频(RF)收发器部件、全球定位系统(GPS)接收器部件和/或其它部件。在一些实施例中，作为无线接口的附加或者替代，通信子系统3224可以提供有线网络连接(例如，以太网)。

[0201] 在一些实施例中，通信子系统3224还可以代表可以使用计算机系统3200的一个或多个用户接收结构化和/或非结构化数据馈送3226、事件流3228、事件更新3230等形式的输入通信。

[0202] 作为示例，通信子系统3224可被配置为实时地从社交网络的用户和/或其它通信服务接收数据馈送3226，诸如**Twitter®**馈送、**Facebook®**更新、诸如丰富站点摘要(RSS)馈送之类的web馈送和/或来自一个或多个第三方信息源的实时更新。

[0203] 此外，通信子系统3224还可被配置为接收连续数据流形式的数据，这可以包括本质上可以是连续的或无界的没有明确终止的实时事件的事件流3228和/或事件更新3230。产生连续数据的应用的示例可以包括，例如，传感器数据应用、金融报价机、网络性能测量

工具(例如,网络监视和业务管理应用)、点击流分析工具、汽车交通监视,等等。

[0204] 通信子系统3224还可被配置为向一个或多个数据库输出结构化和/或非结构化的数据馈送3226、事件流3228、事件更新3230,等等,这一个或多个数据库可以与耦合到计算机系统3200的一个或多个流式数据源计算机通信。

[0205] 计算机系统3200可以是各种类型之一,包括手持便携式设备(例如,iPhone®蜂窝电话、iPad®计算平板电脑、PDA)、可穿戴设备(例如,Google Glass®头戴式显示器)、PC、工作站、大型机、信息站、服务器机架、或任何其它数据处理系统。

[0206] 由于计算机和网络的不断变化的本质,在图中绘出的计算机系统3200的描述仅仅作为具体的示例。具有比图中绘出的系统更多或更少部件的许多其它配置是可能的。例如,定制的硬件也可以被使用和/或特定的元素可以用硬件、固件、软件(包括小程序)或其组合来实现。另外,也可以采用到诸如网络输入/输出设备之类的其它计算设备的连接。基于本文提供的公开内容和教导,本领域普通技术人员将认识到实现各种实施例的其它方式和/或方法。

[0207] 在前面的描述中,出于解释的目的,阐述了许多具体细节以便提供对本发明的各种实施例的透彻理解。然而,对于本领域技术人员清楚的是,可以在没有这些具体细节中的一些的情况下实践本发明的实施例。在其它情况下,以框图形式示出了众所周知的结构和设备。

[0208] 以上描述仅提供示例性实施例,并且不旨在限制本公开的范围、适用性或配置。更确切地说,示例性实施例的前述描述将为本领域技术人员提供用于实现示例性实施例的使能描述。应当理解的是,在不脱离所附权利要求中阐述的本发明的精神和范围的情况下,可以对元件的功能和布置进行各种改变。

[0209] 在前面的描述中给出了具体细节以提供对实施例的透彻理解。然而,本领域普通技术人员将理解的是,可以在没有这些具体细节的情况下实践这些实施例。例如,电路、系统、网络、过程和其它部件可能已经以框图形式示出为部件,以免用不必要的细节模糊实施例。在其它情况下,可能已经示出了众所周知的电路、过程、算法、结构和技术而没有不必要的细节,以避免模糊实施例。

[0210] 而且,要注意的是,各个实施例可能已被描述为过程,该过程被描绘为流程图、流程示意图、结构图或框图。虽然流程图可能已将操作描述为顺序过程,但许多操作可以并行或同时执行。此外,可以重新布置操作的次序。过程在其操作完成时终止,但可以有未包含在图中的附加步骤。过程可以与方法、函数、程序、子例程、子程序等对应。当过程与函数对应时,其终止可以与函数返回到调用函数或主函数对应。

[0211] 术语“计算机可读介质”包括但不限于便携式或固定存储设备、光学存储设备、无线信道以及能够存储、包含或携带(一个或多个)指令和/或数据的各种其它介质。代码段或机器可执行指令可以表示过程、函数、子程序、程序、例程、子例程、模块、软件包、类或指令、数据结构或程序语句的任何组合。代码段可以通过传递和/或接收信息、数据、自变量、参数或存储器内容而耦合到另一个代码段或硬件电路。信息、自变量、参数、数据等可以经由任何合适的手段传递、转发或发送,这些手段包括存储器共享、消息传递、令牌传递、网络传输等。

[0212] 此外,实施例可以通过硬件、软件、固件、中间件、微代码、硬件描述语言或其任何

组合来实现。当在软件、固件、中间件或微代码中实现时，用于执行必要任务的程序代码或代码段可以存储在机器可读介质中。(一个或多个)处理器可以执行必要的任务。

[0213] 在前述说明书中，参考其具体实施例对本发明的各方面进行了描述，但是本领域技术人员将认识到的是，本发明不限于此。上述公开的各个特征和方面可以被单独使用或联合使用。另外，在不脱离本说明书的更广泛精神和范围的情况下，实施例可以在除本文所述的环境和应用之外的任何数目的环境和应用中被使用。相应地，本说明书和附图应当被认为是说明性的而不是限制性的。

[0214] 此外，出于说明的目的，以特定顺序描述了方法。应该认识到的是，在替代实施例中，可以以与所描述的顺序不同的顺序执行方法。还应该认识到的是，上述方法可以由硬件部件执行，或者可以以机器可执行指令的序列被实施，机器可执行指令可以用于使机器(诸如编程有指令的通用或专用处理器或逻辑电路)执行方法。这些机器可执行指令可以被存储在一个或多个机器可读介质上，诸如CD-ROM或其它类型的光盘、软盘、ROM、RAM、EPROM、EEPROM、磁卡或光卡、闪存、或适合于存储电子指令的其它类型的机器可读介质。替代地，可以通过硬件和软件的组合来执行方法。

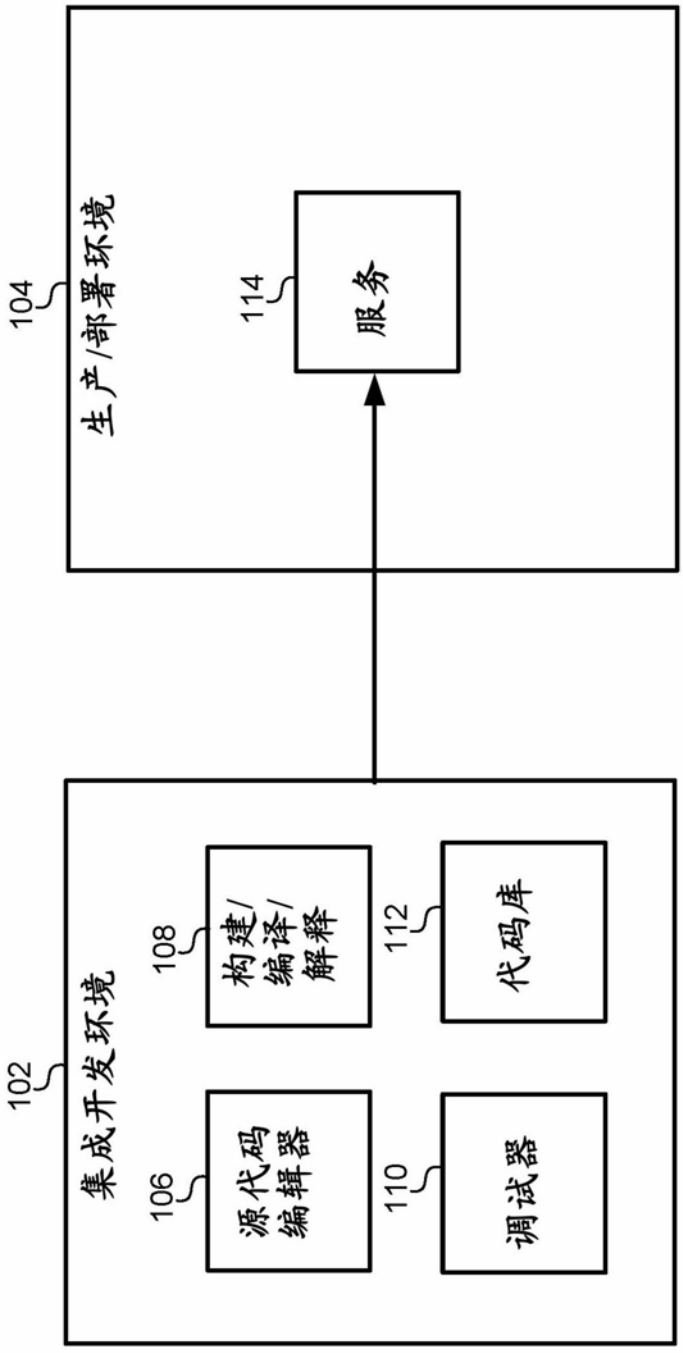

104
生产/部署环境
114
服务
102
集成开发环境
108
构建/编译/解释
112
代码库
106
源代码编辑器
110
调试器

图1

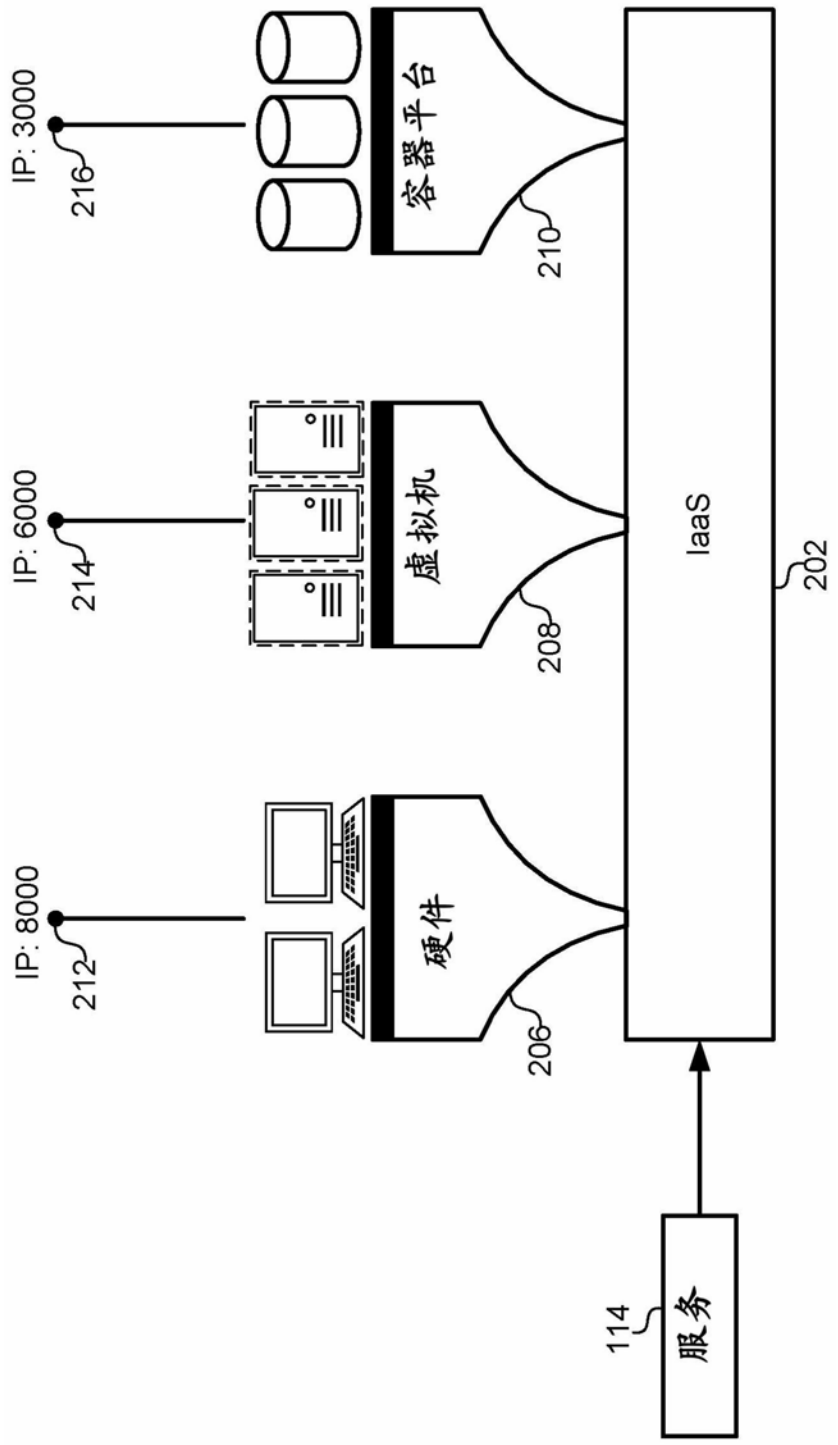

IP: 3000
216
容器平台
210
IP: 6000
214
虚拟机
208
IaaS
202
IP: 8000
212
硬件
206
114
服务

图2

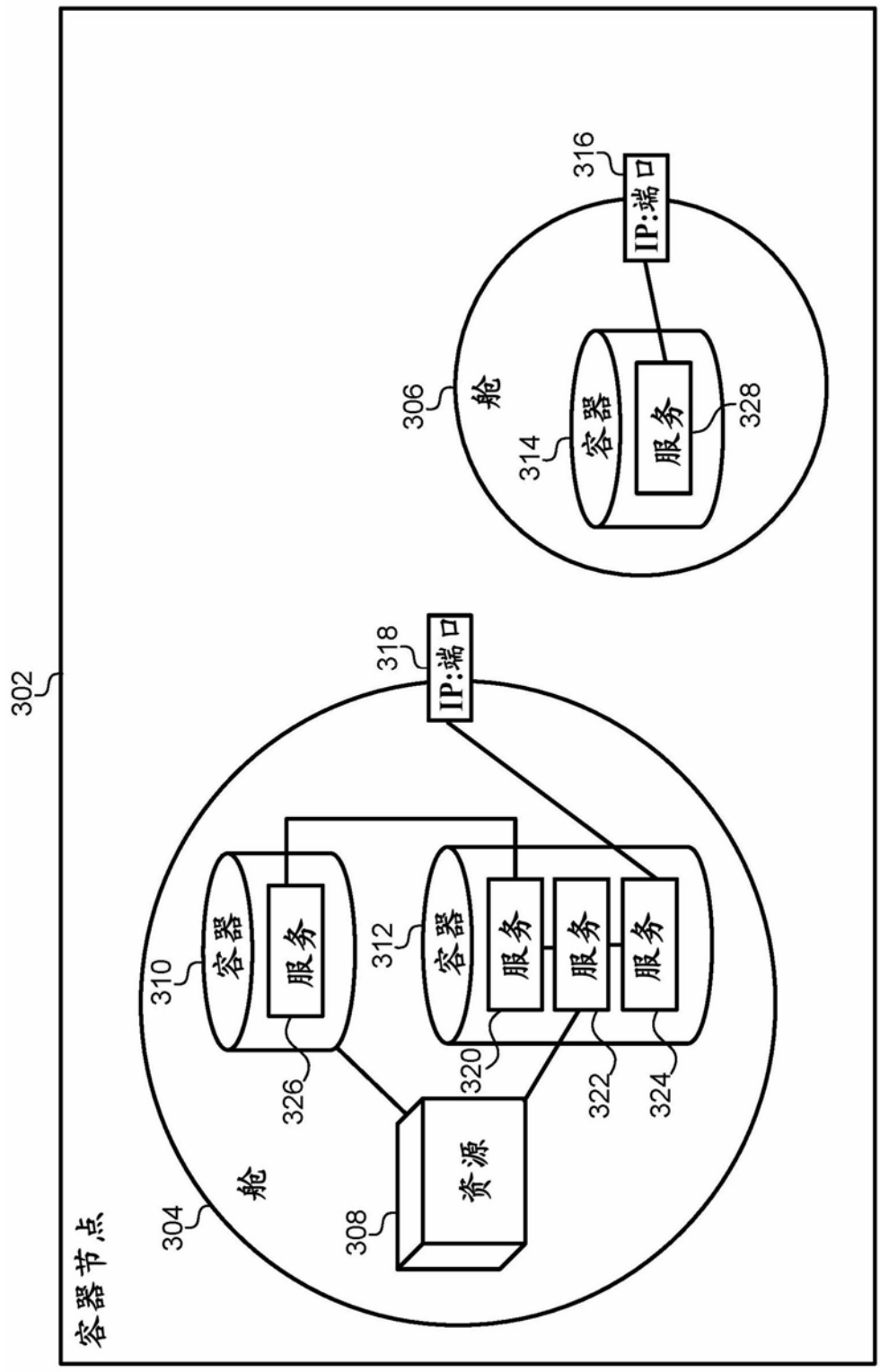

容器节点
302
舱
304
308
资源
310
容器
服务
326
312
容器
服务
服务
服务
320
322
324
318
IP:端口
舱
306
316
IP:端口
314
容器
服务
328

图3

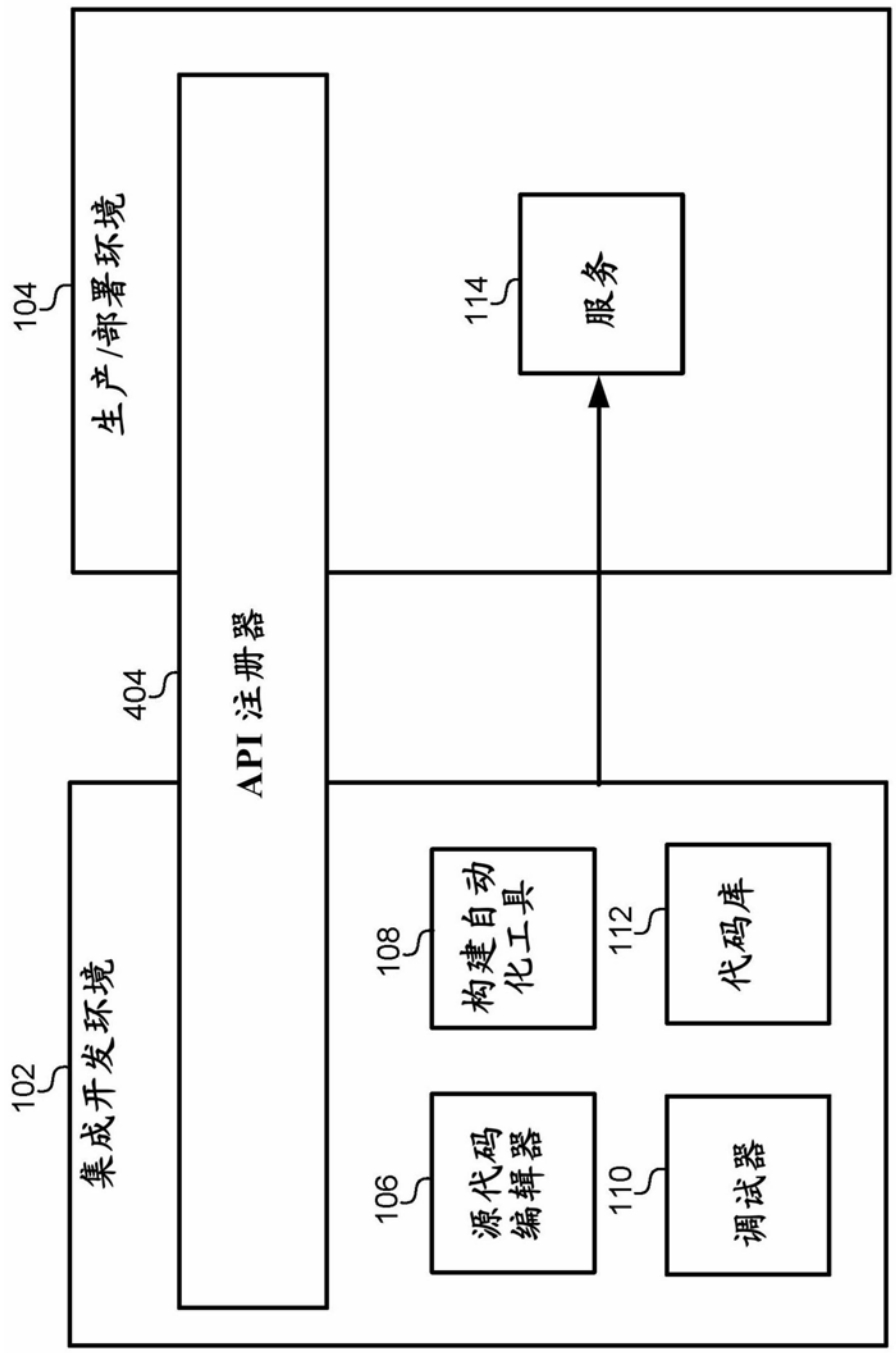

104
生产/部署环境
114
服务
404
API 注册器
102
集成开发环境
108
构建自动化工具
112
代码库
106
源代码编辑器
110
调试器

图4

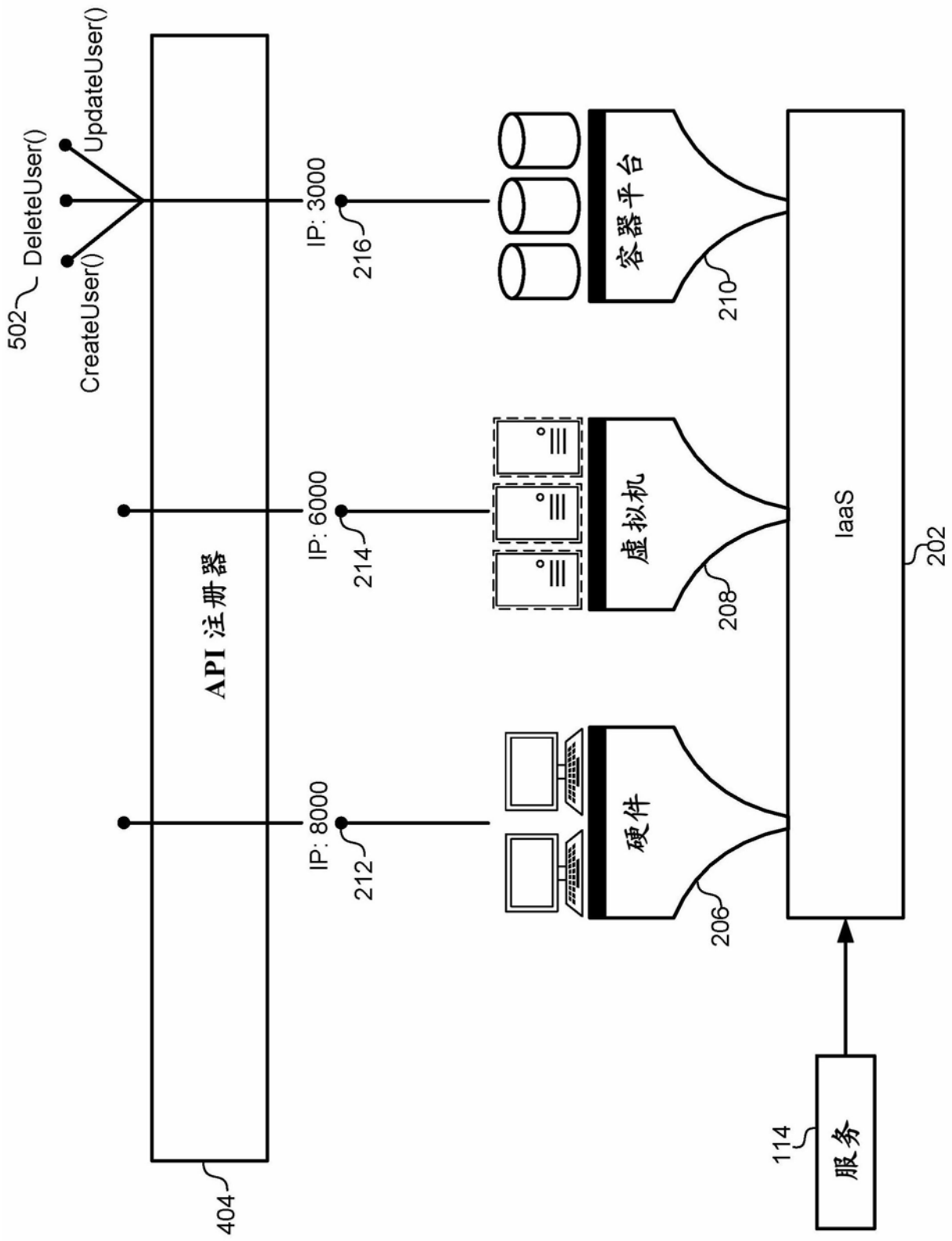

UpdateUser()
DeleteUser()
CreateUser()
502
API 注册器
404
IP: 3000
216
IP: 6000
214
IP: 8000
212
容器平台
210
虚拟机
208
硬件
206
IaaS
202
服务
114

图5

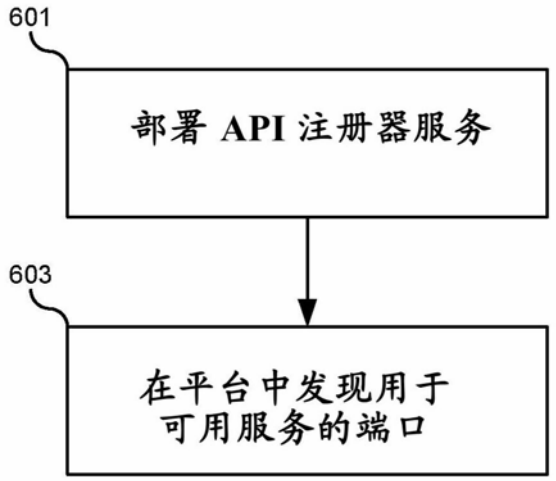


图6A

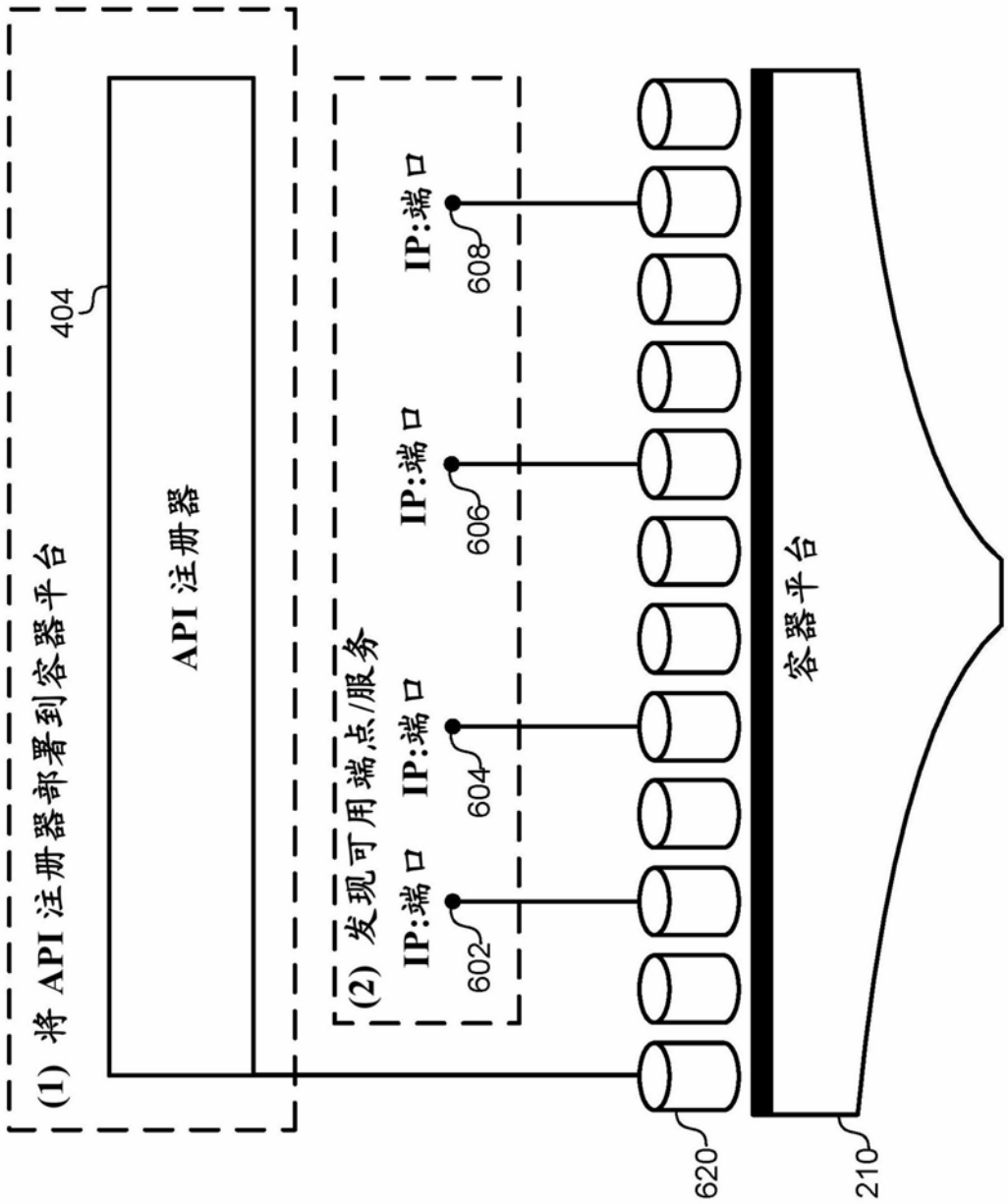

(1) 将 API 注册器部署到容器平台
404
API 注册器
(2) 发现可用端点/服务
IP:端口
602
IP:端口
604
IP:端口
606
IP:端口
608
容器平台
620
210

图6B

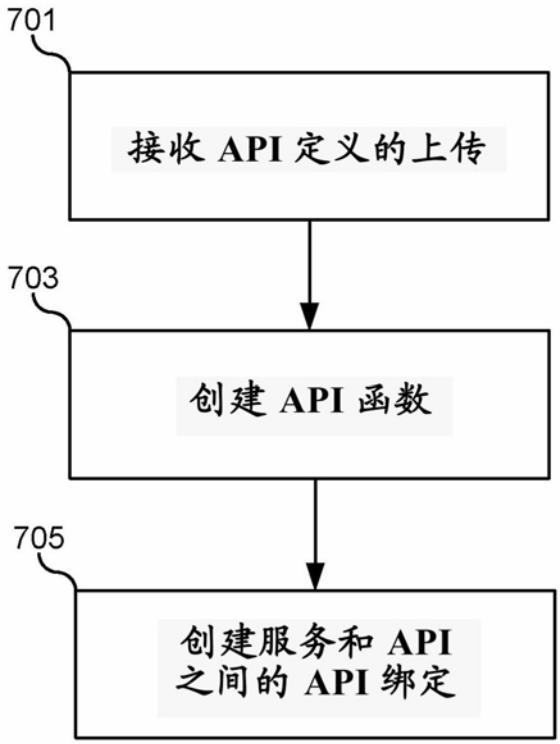


图7A

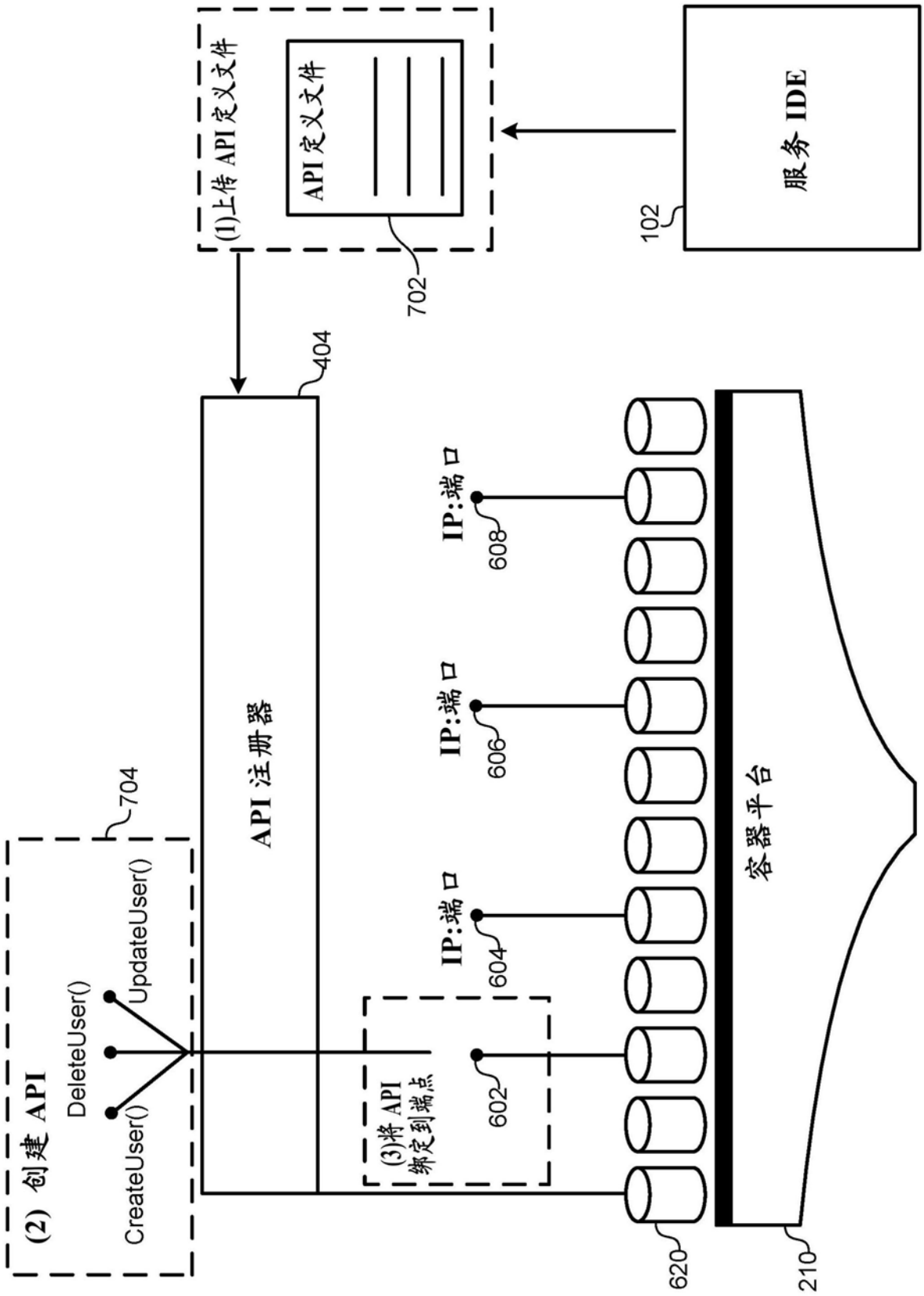

(1)上传 API 定义文件
API 定义文件
702
102
服务 IDE
404
API 注册器
704
(2) 创建 API
CreateUser()
DeleteUser()
UpdateUser()
(3)将 API 绑定到端点
602
IP:端口
604
IP:端口
606
IP:端口
608
容器平台
620
210

图7B

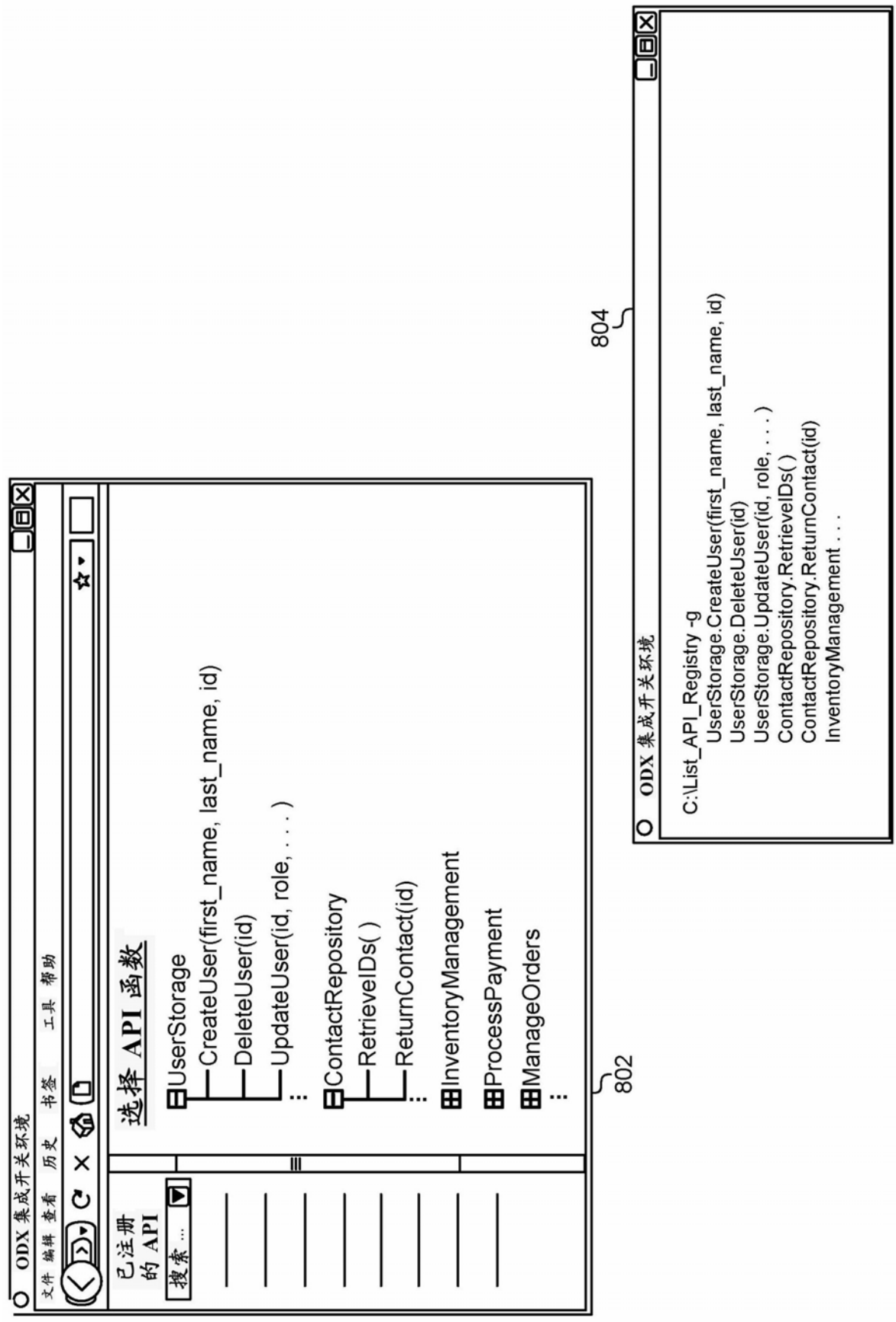

ODX 集成开关环境
文件 编辑 查看 历史 书签 工具 帮助
已注册的 API
搜索...
选择 API 函数
UserStorage
CreateUser(first_name, last_name, id)
DeleteUser(id)
UpdateUser(id, role, . . .)
...
ContactRepository
RetrieveIDs( )
ReturnContact(id)
...
InventoryManagement
ProcessPayment
ManageOrders
...
802
804
ODX 集成开关环境
C:\List_API_Registry -g
UserStorage.CreateUser(first_name, last_name, id)
UserStorage.DeleteUser(id)
UserStorage.UpdateUser(id, role, . . .)
ContactRepository.RetrieveIDs( )
ContactRepository.ReturnContact(id)
InventoryManagement . . .

图8

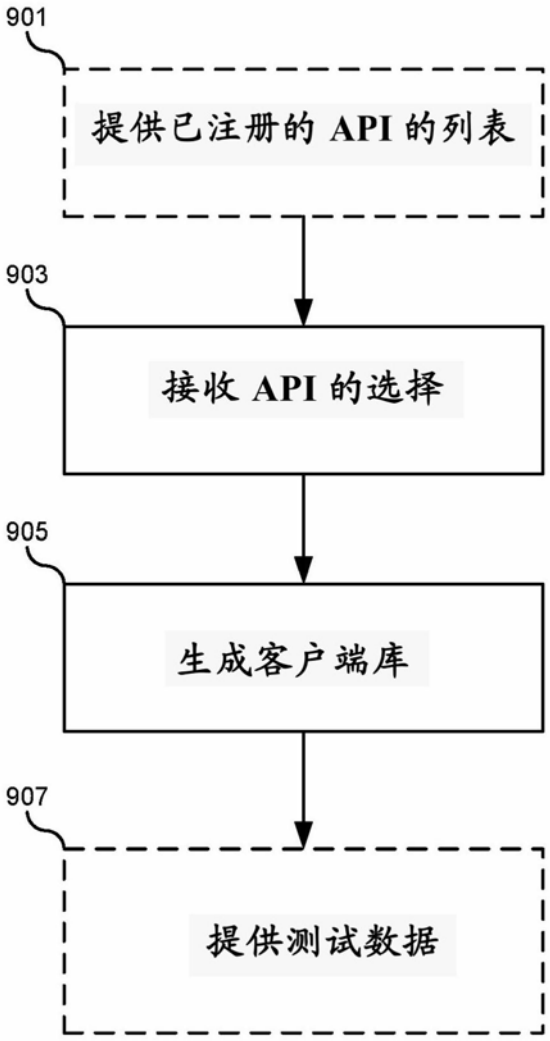


图9

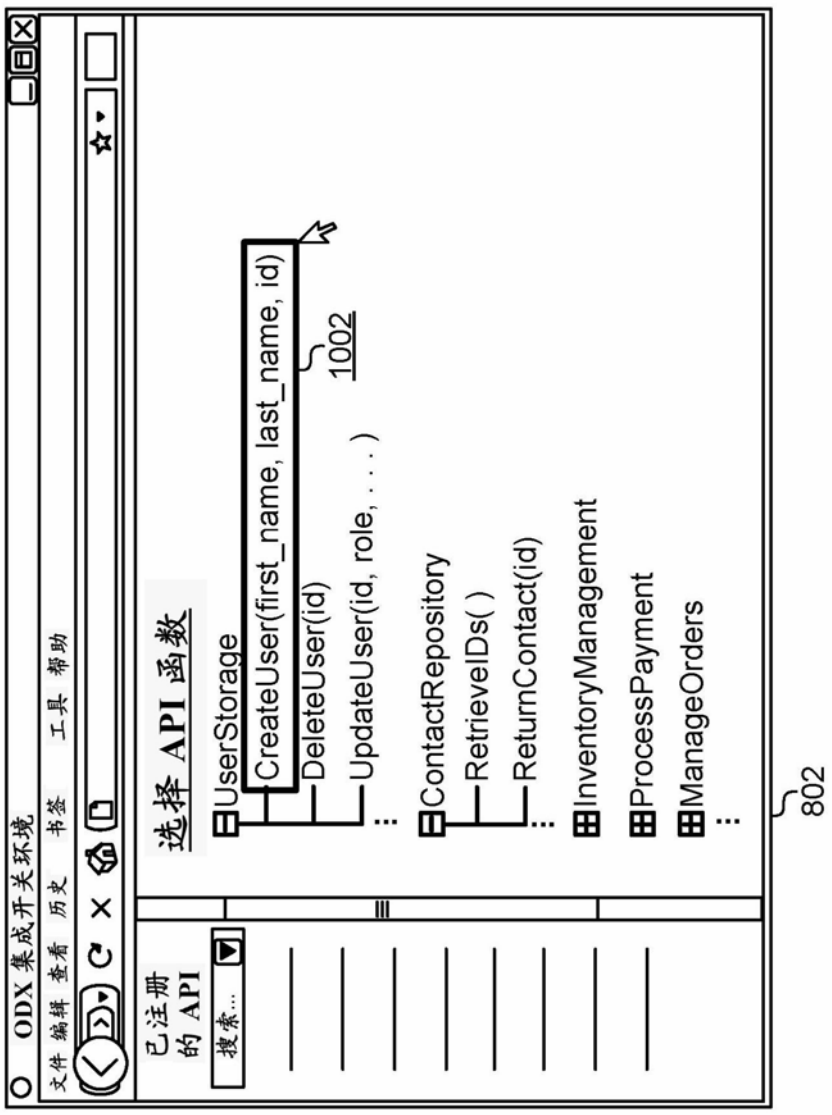

ODX 集成开关环境
文件 编辑 查看 历史 书签 工具 帮助
已注册的 API
搜索...
选择 API 函数
UserStorage
CreateUser(first_name, last_name, id)
DeleteUser(id)
UpdateUser(id, role, . . .)
...
ContactRepository
RetrieveIDs( )
ReturnContact(id)
...
InventoryManagement
ProcessPayment
ManageOrders
...
1002
802

图10

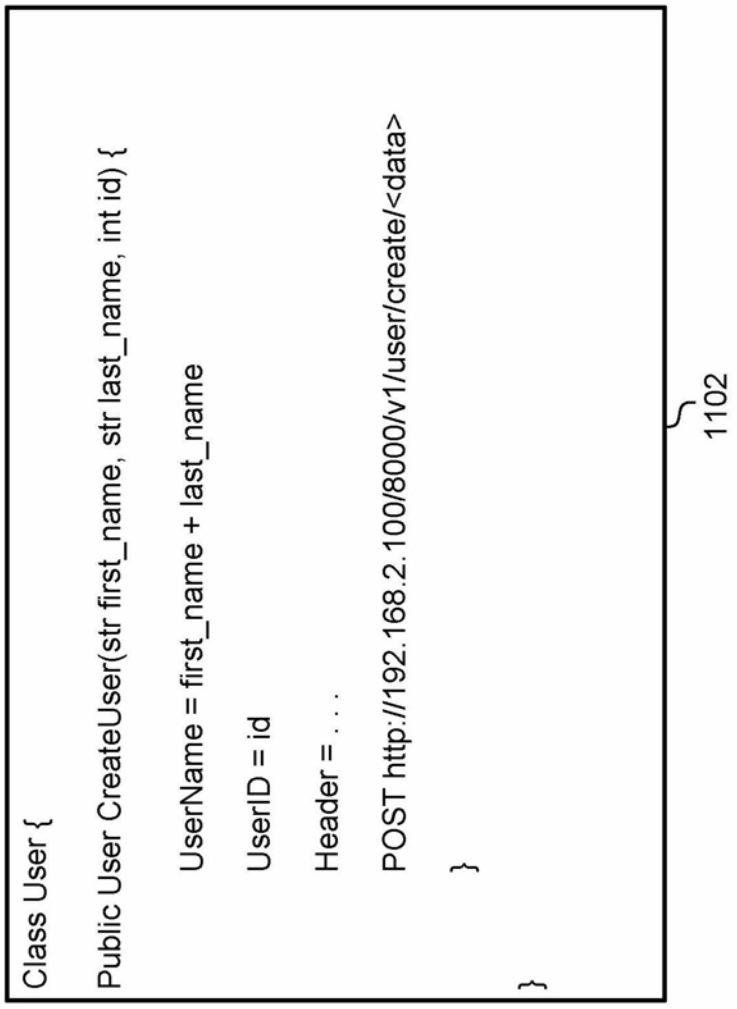

Class User {
Public User CreateUser(str first_name, str last_name, int id) {
UserName = first_name + last_name
UserID = id
Header = . . .
POST http://192.168.2.100/8000/v1/user/create/<data>
}
}
1102

图11

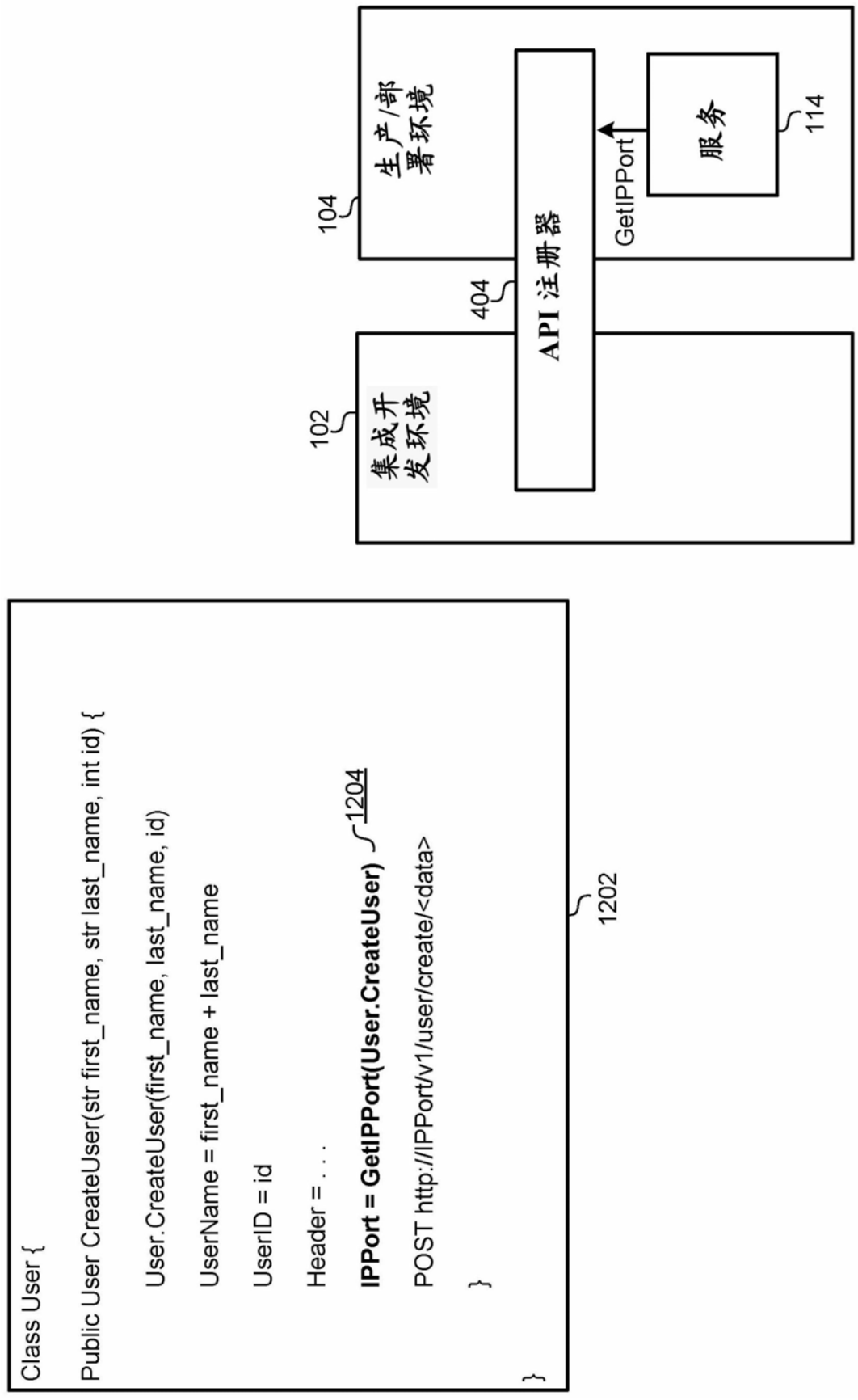

Class User {
Public User CreateUser(str first_name, str last_name, int id) {
User.CreateUser(first_name, last_name, id)
UserName = first_name + last_name
UserID = id
Header = . . .
IPPort = GetIPPort(User.CreateUser)
POST http://IPPort/v1/user/create/<data>
}
}
1202
1204
102
集成开发环境
404
API 注册器
104
生产/部署环境
GetIPPort
服务
114

图12

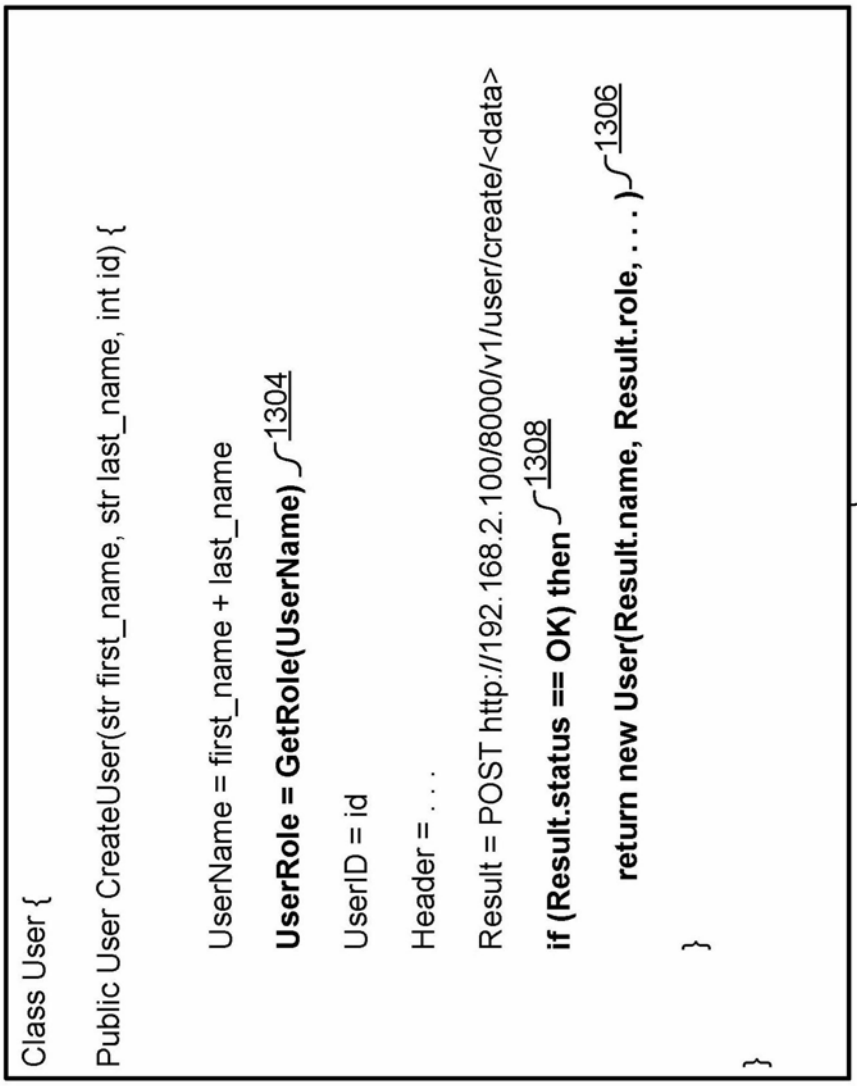

Class User {
Public User CreateUser(str first_name, str last_name, int id) {
UserName = first_name + last_name
UserRole = GetRole(UserName) 1304
UserID = id
Header = . . .
Result = POST http://192.168.2.100/8000/v1/user/create/<data>
if (Result.status == OK) then 1308
return new User(Result.name, Result.role, . . .) 1306
}
}
1302

图13

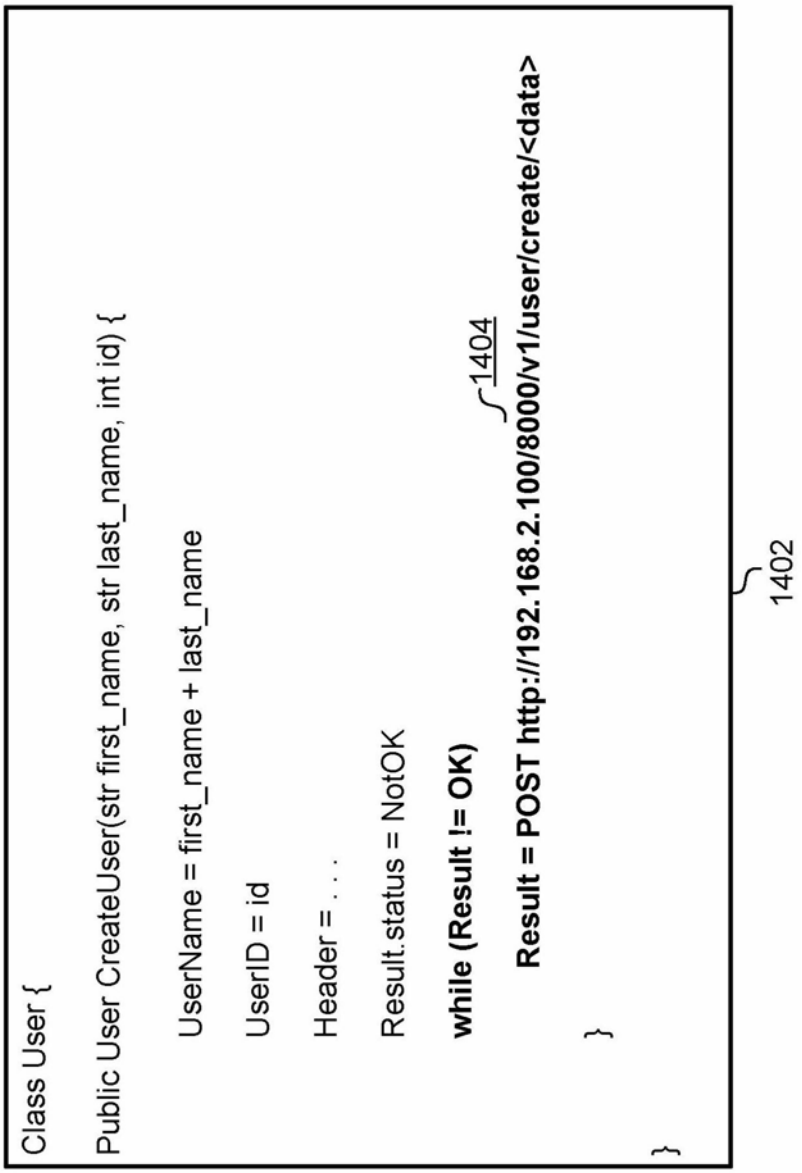

Class User {
Public User CreateUser(str first_name, str last_name, int id) {
UserName = first_name + last_name
UserID = id
Header = . . .
Result.status = NotOK
while (Result != OK)
Result = POST http://192.168.2.100/8000/v1/user/create/<data>
}
}
1404
1402

图14

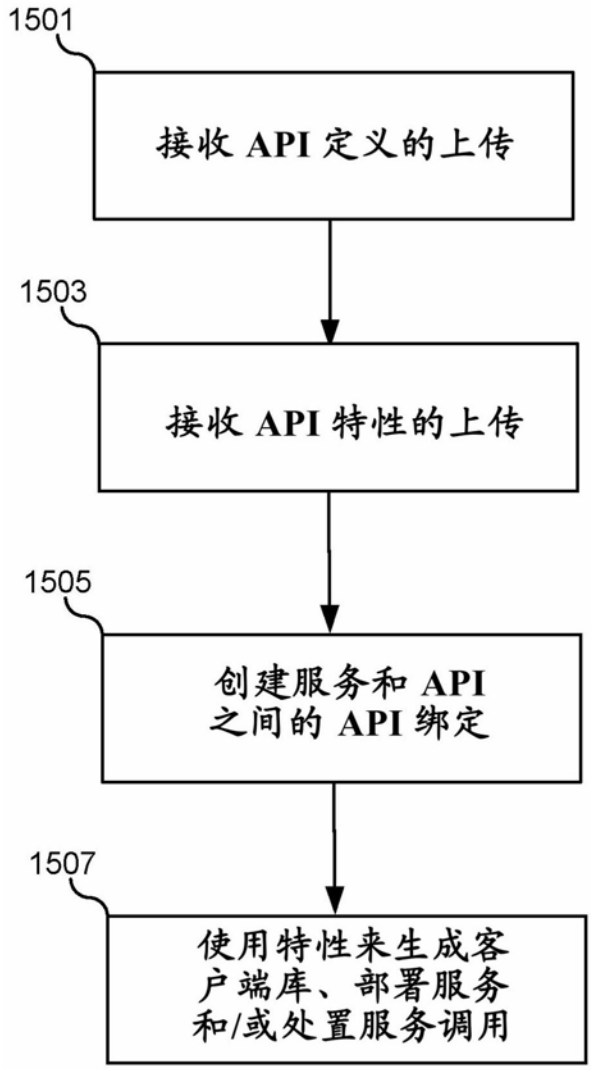

1501
接收 API 定义的上传
1503
接收 API 特性的上传
1505
创建服务和 API
之间的 API 绑定
1507
使用特性来生成客
户端库、部署服务
和/或处置服务调用

图15A

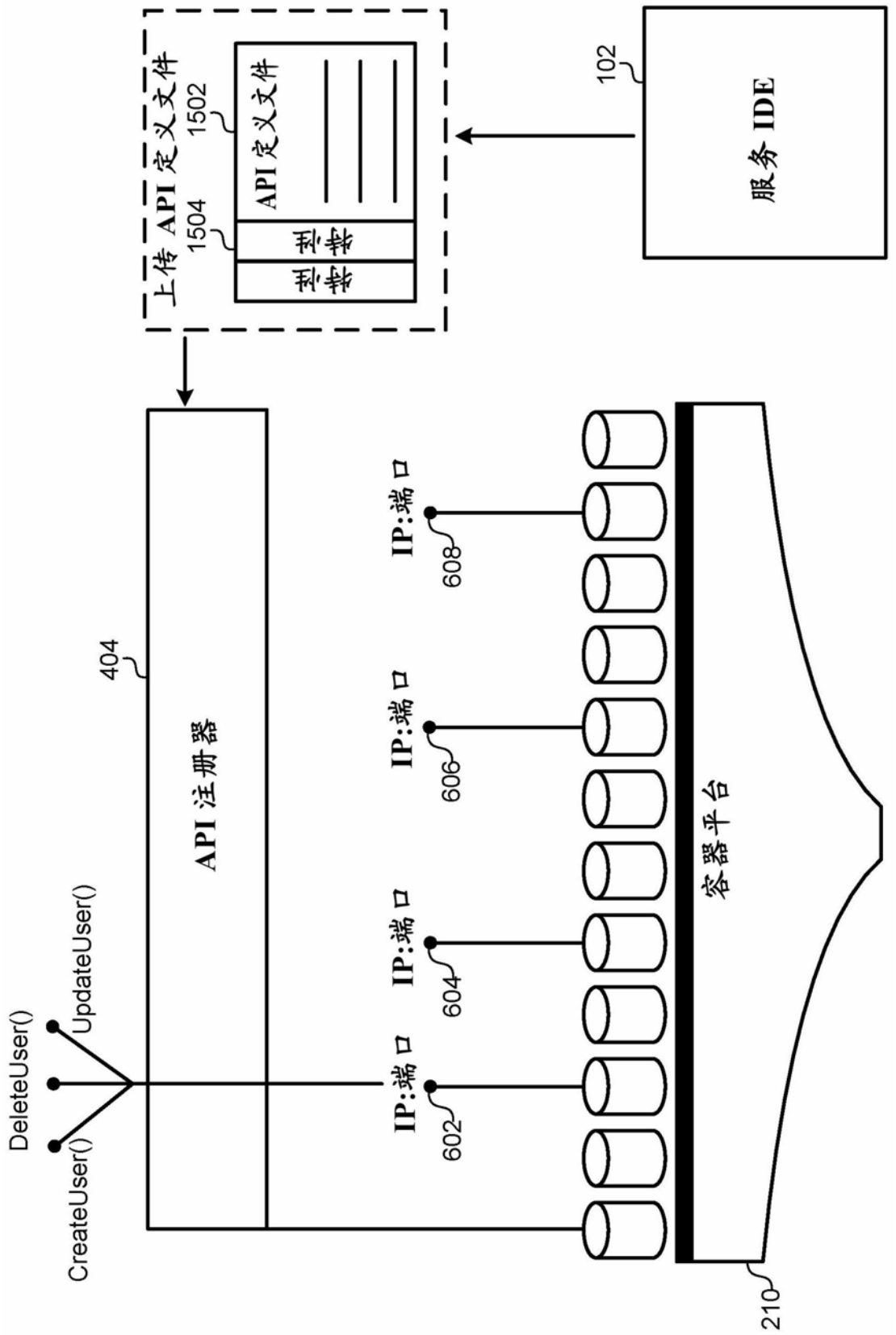

服务 IDE
102
上传 API 定义文件
API 定义文件
1502
1504
404
API 注册器
DeleteUser()
UpdateUser()
CreateUser()
IP:端口
602
IP:端口
604
IP:端口
606
IP:端口
608
容器平台
210

图15B

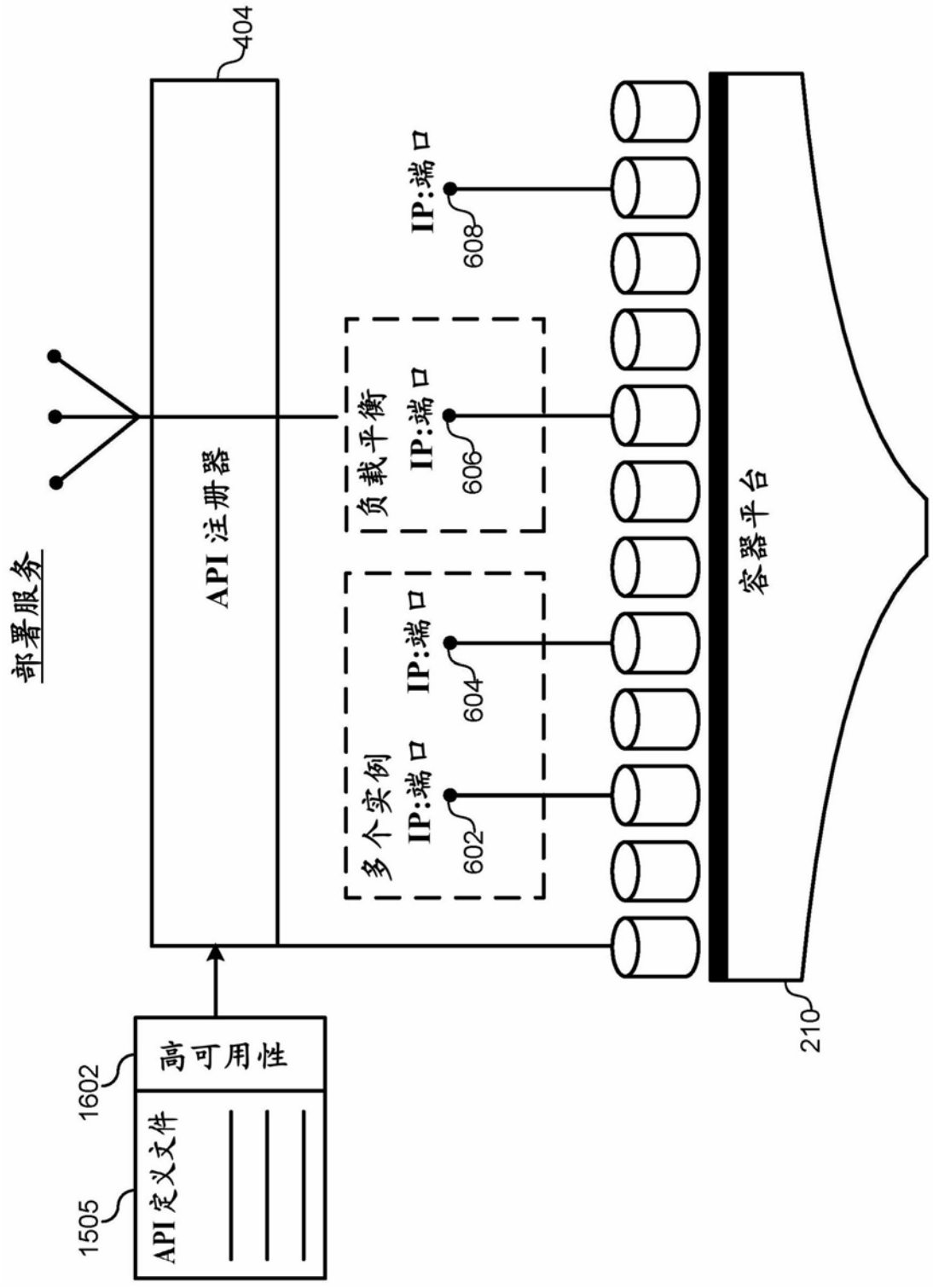

404
部署服务
API 注册器
负载平衡
多个实例
IP:端口
602
IP:端口
604
IP:端口
606
IP:端口
608
容器平台
210
高可用性
1602
API 定义文件
1505

图16

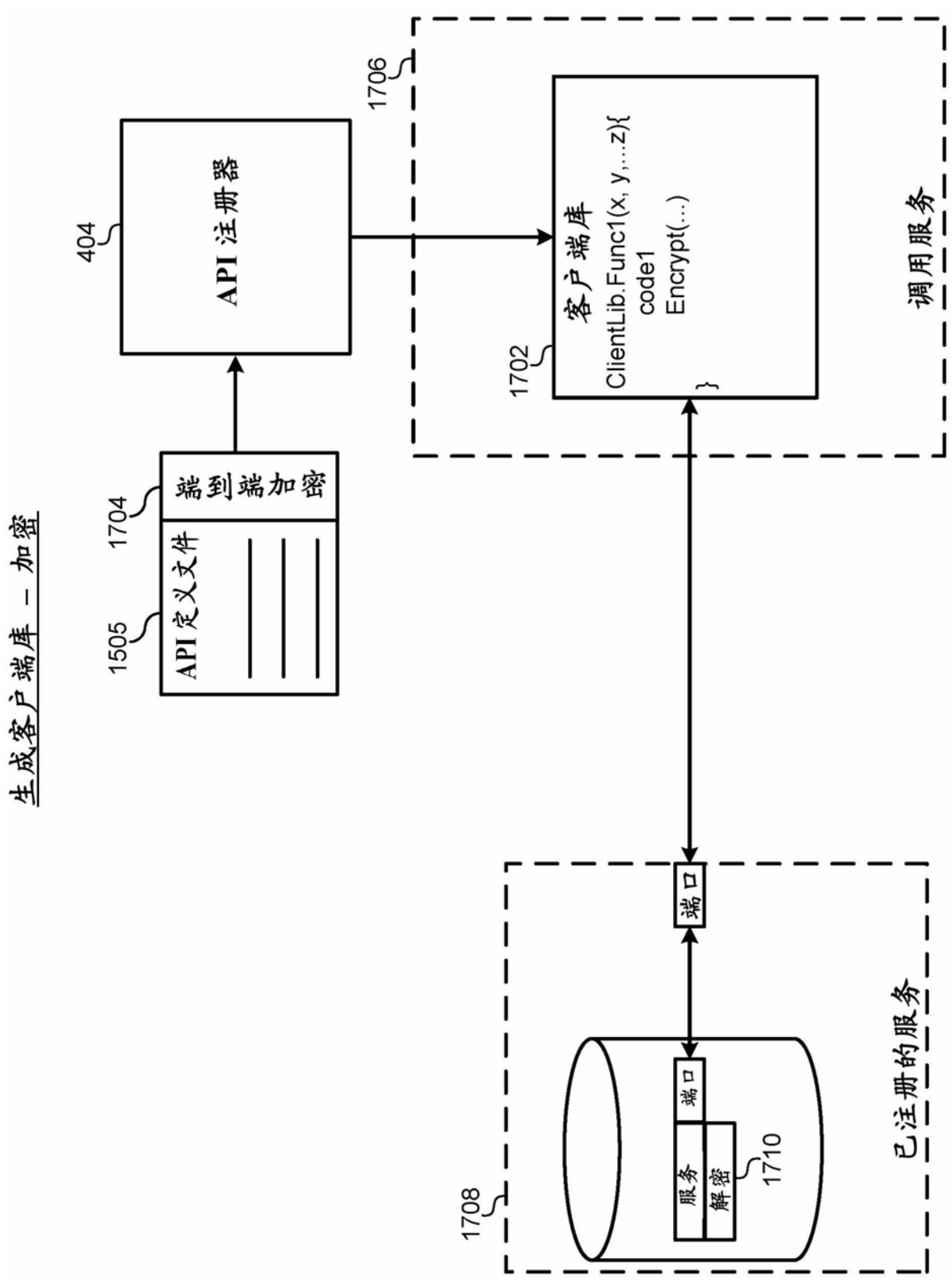

生成客户端库 – 加密
API 定义文件
端到端加密
1505
1704
API 注册器
404
1706
客户端库
ClientLib.Func1(x, y,...z){
code1
Encrypt(...)
}
1702
调用服务
端口
服务
解密
1710
1708
已注册的服务

图17

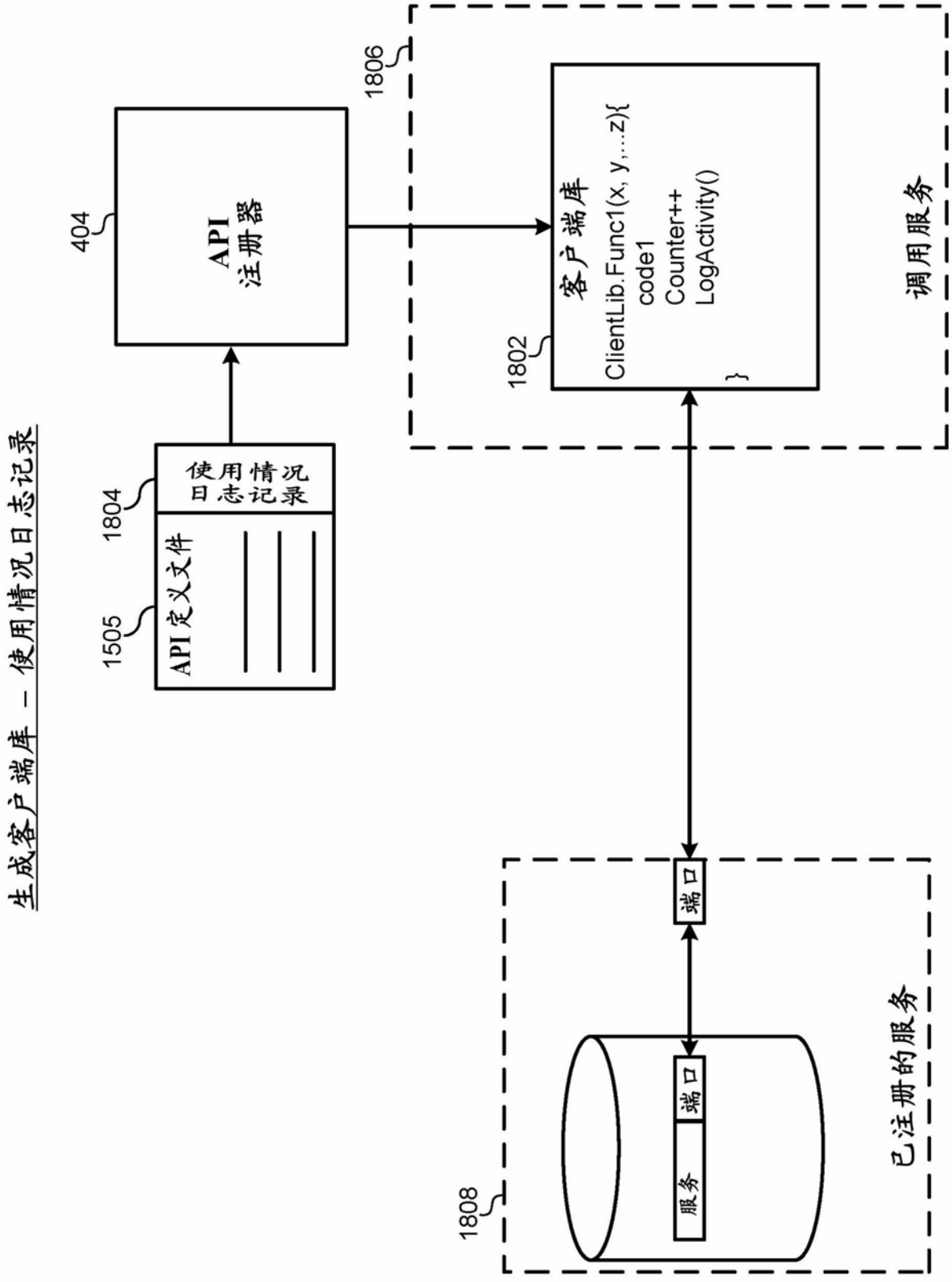

生成客户端库 – 使用情况日志记录
404
API
注册器
1806
客户端库
ClientLib.Func1(x, y,...z){
code1
Counter++
LogActivity()
}
1802
调用服务
1804
使用情况
日志记录
1505
API 定义文件
端口
端口
服务
1808
已注册的服务

图18

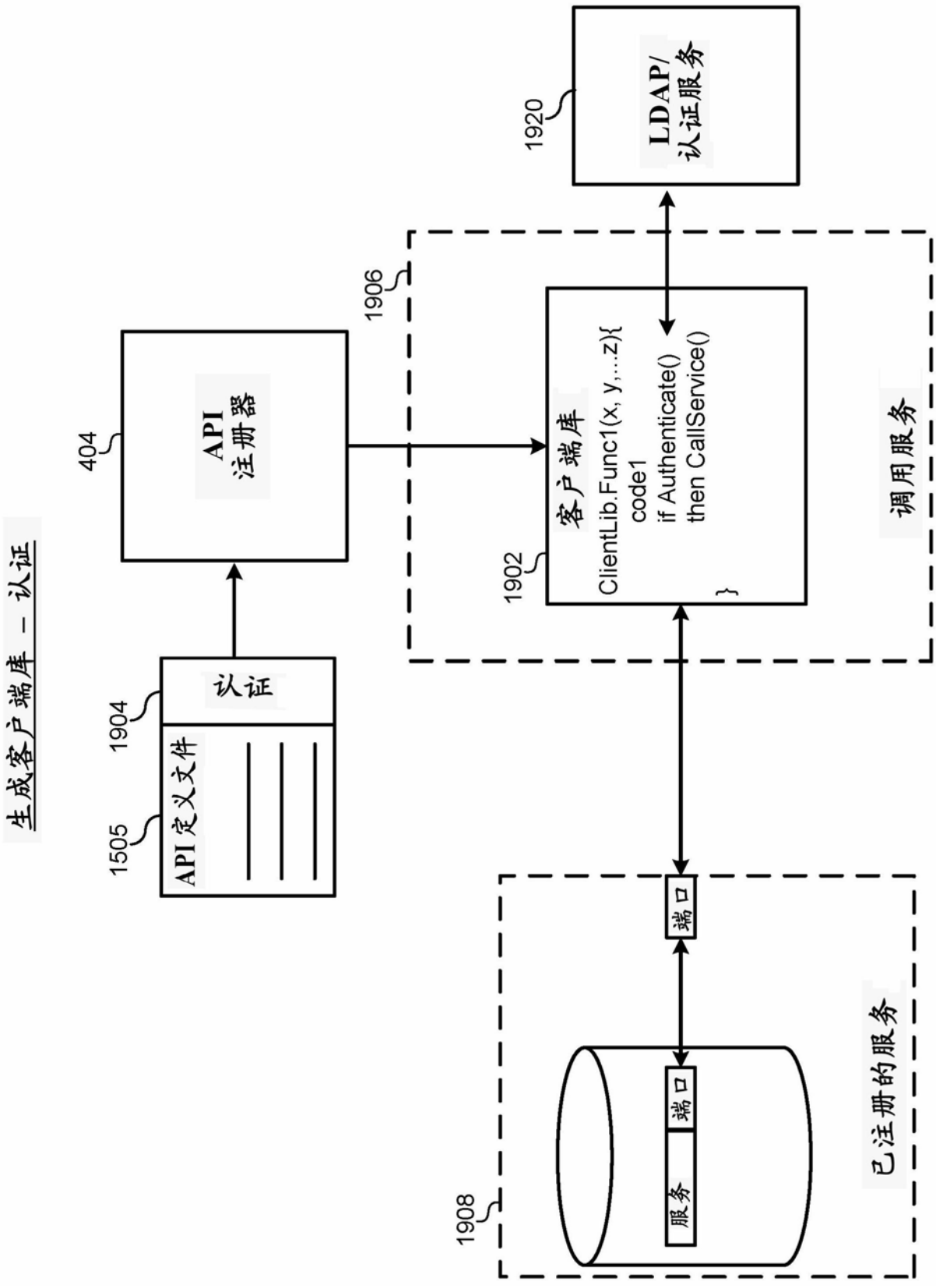

生成客户端库 – 认证
1505
API 定义文件
1904
认证
404
API
注册器
1906
1902
客户端库
ClientLib.Func1(x, y,...z){
code1
if Authenticate()
then CallService()
}
调用服务
1920
LDAP/
认证服务
1908
服务
端口
端口
已注册的服务

图19

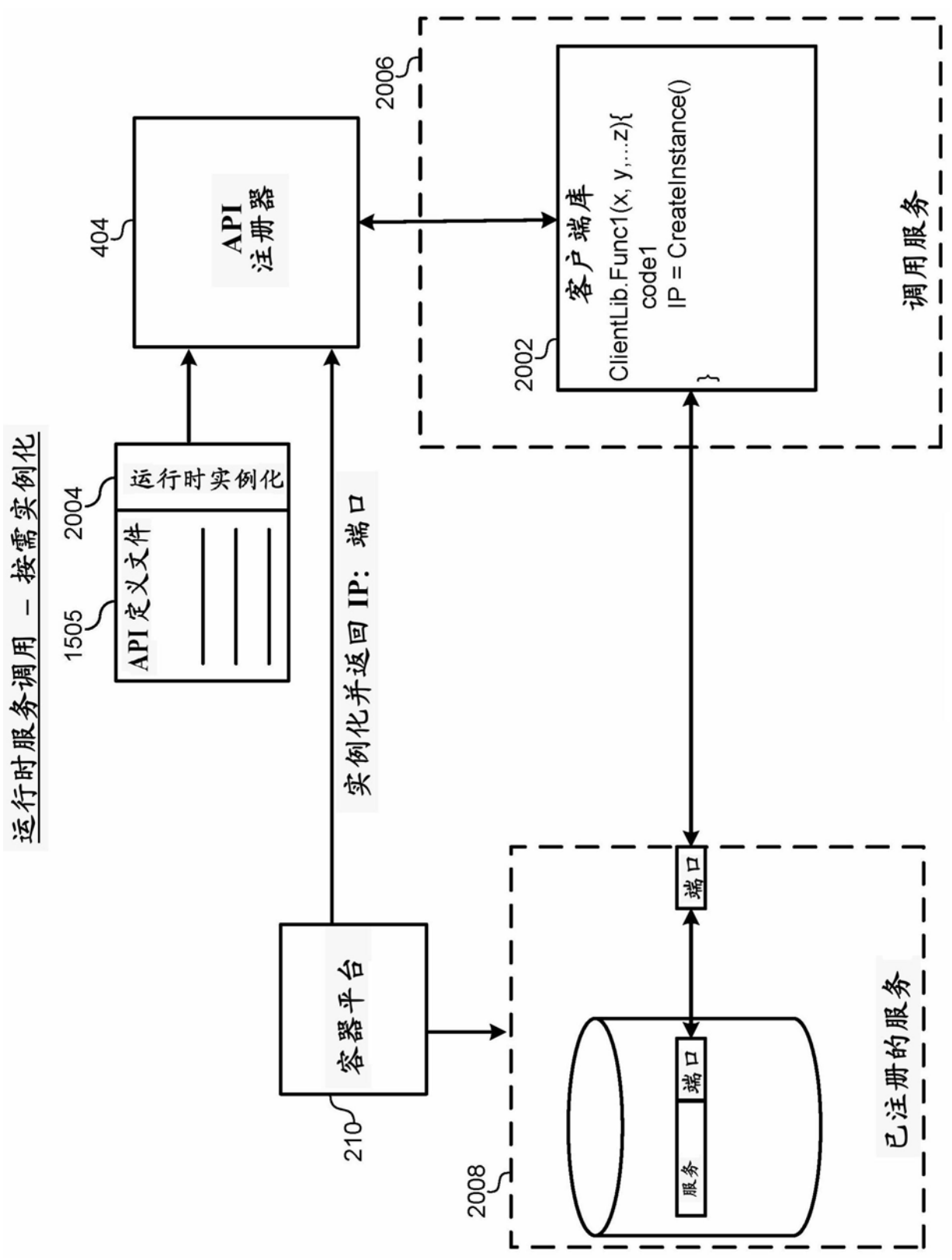

运行时服务调用 – 按需实例化
404
API
注册器
2006
客户端库
ClientLib.Func1(x, y,...z){
code1
IP = CreateInstance()
}
2002
调用服务
2004
运行时实例化
1505
API定义文件
实例化并返回 IP：端口
端口
容器平台
210
端口
服务
2008
已注册的服务

图20

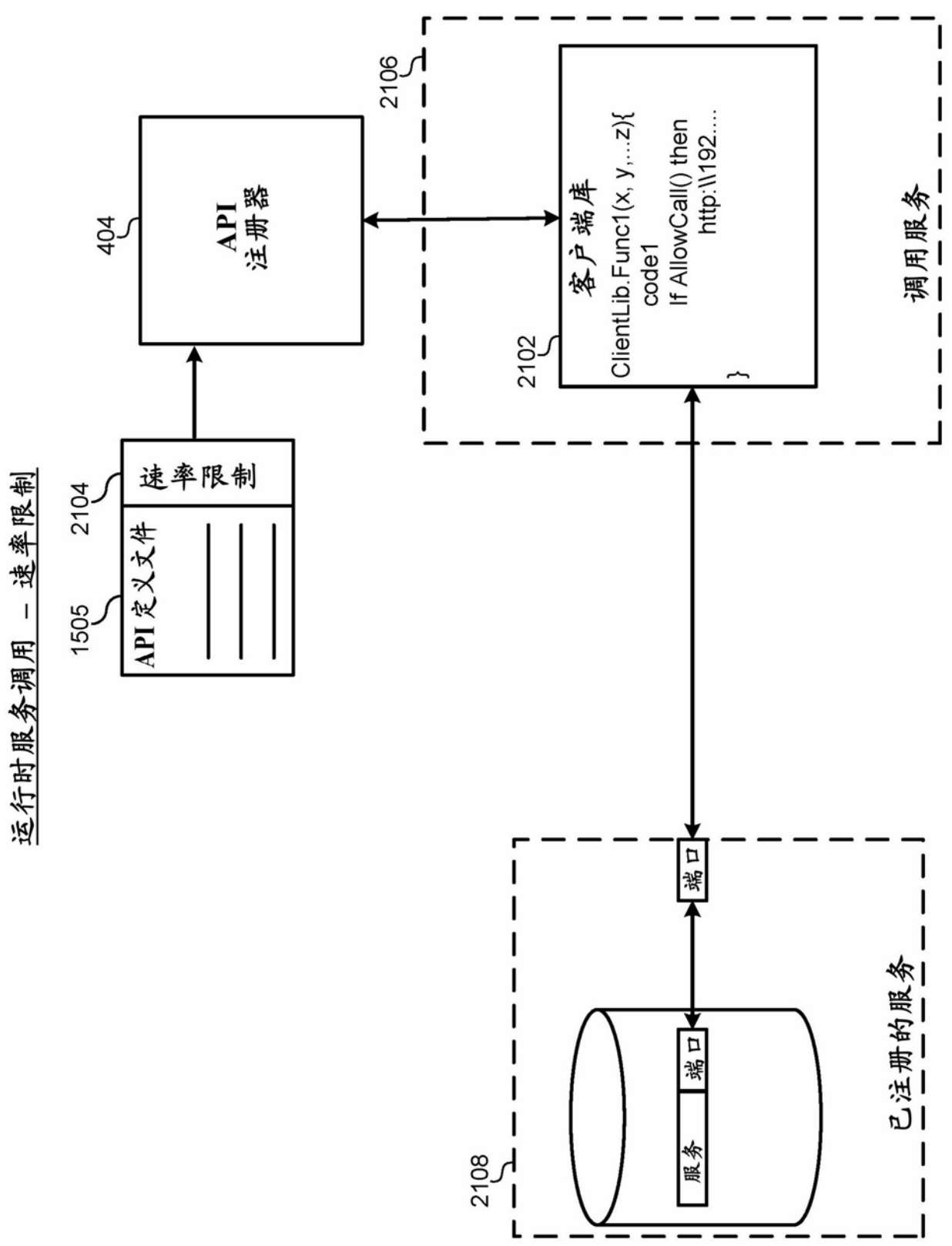

运行时服务调用 – 速率限制
404
API
注册器
2106
客户端库
ClientLib.Func1(x, y,...z){
code1
If AllowCall() then
http:\\192....
}
2102
调用服务
速率限制
2104
API 定义文件
1505
端口
端口
服务
2108
已注册的服务

图21

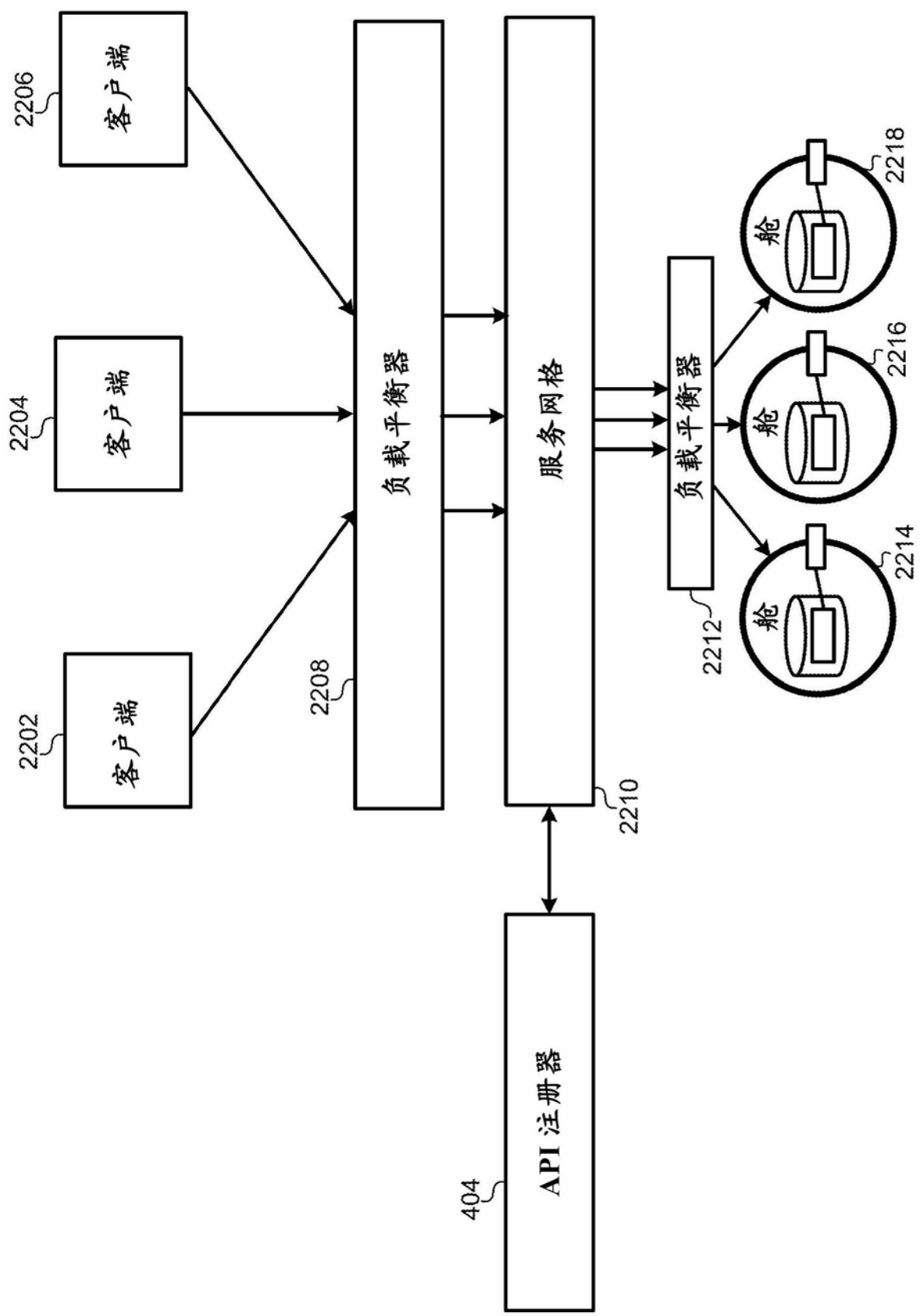

客户端
2206
客户端
2204
客户端
2202
负载平衡器
2208
服务网格
2210
负载平衡器
2212
舱
2218
舱
2216
舱
2214
API 注册器
404

图22

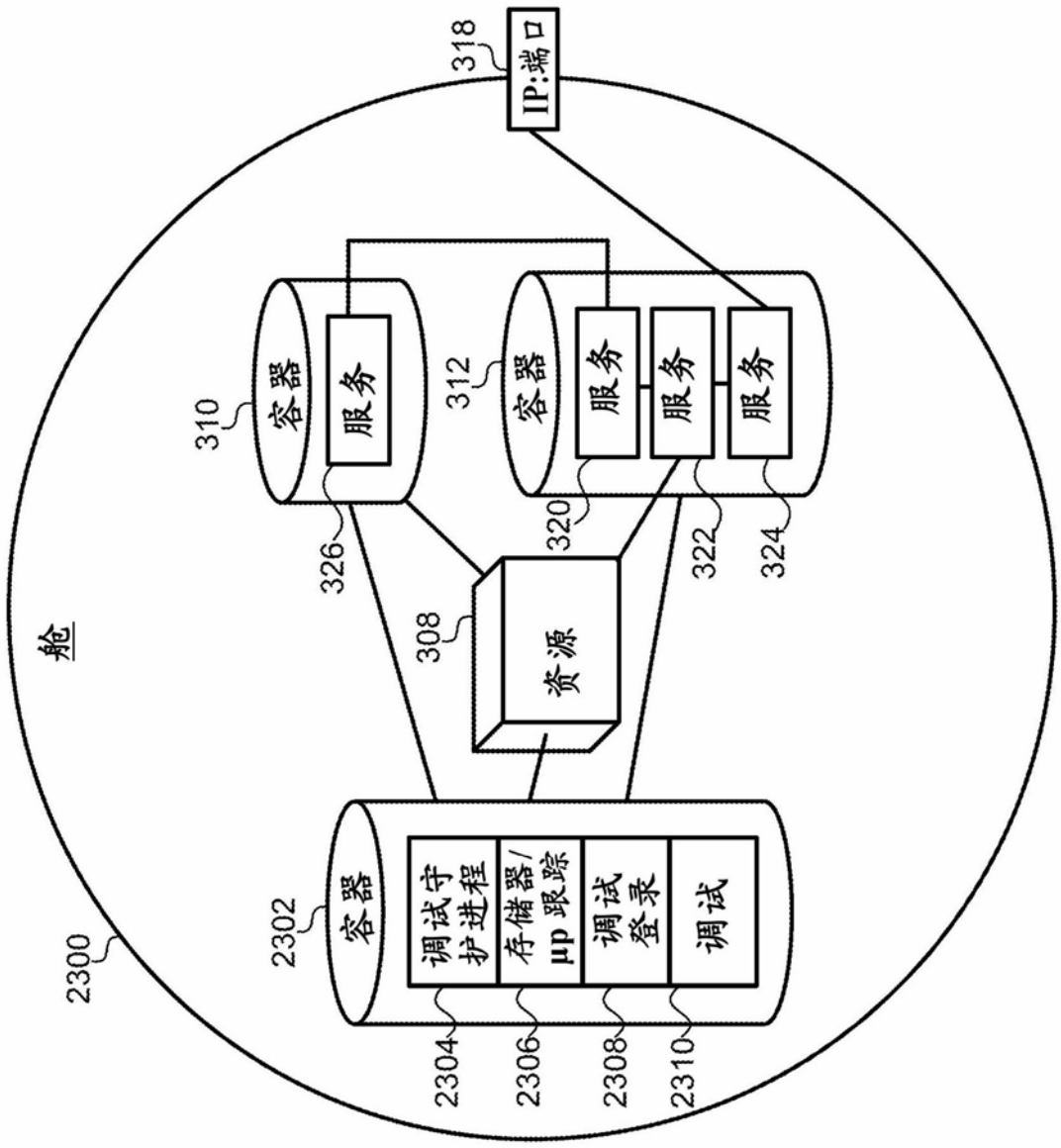

舱
2300
IP:端口
318
容器
服务
310
容器
服务
服务
服务
312
326
320
322
324
资源
308
容器
调试守护进程
存储器/μp跟踪
调试登录
调试
2302
2304
2306
2308
2310

图23

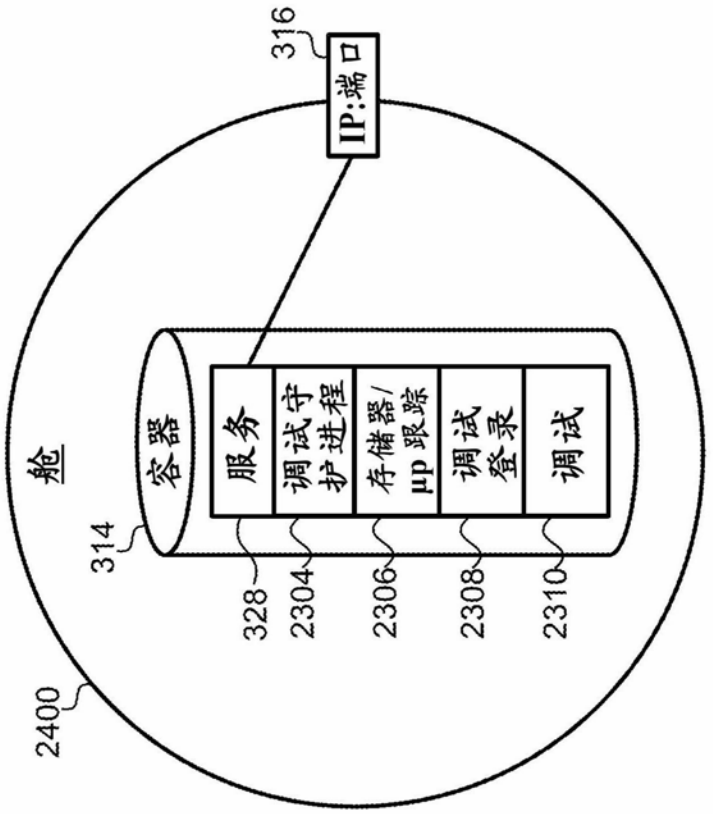

316
IP:端口
舱
容器
服务
调试守护进程
存储器/μp跟踪
调试登录
调试
314
328
2304
2306
2308
2310
2400

图24

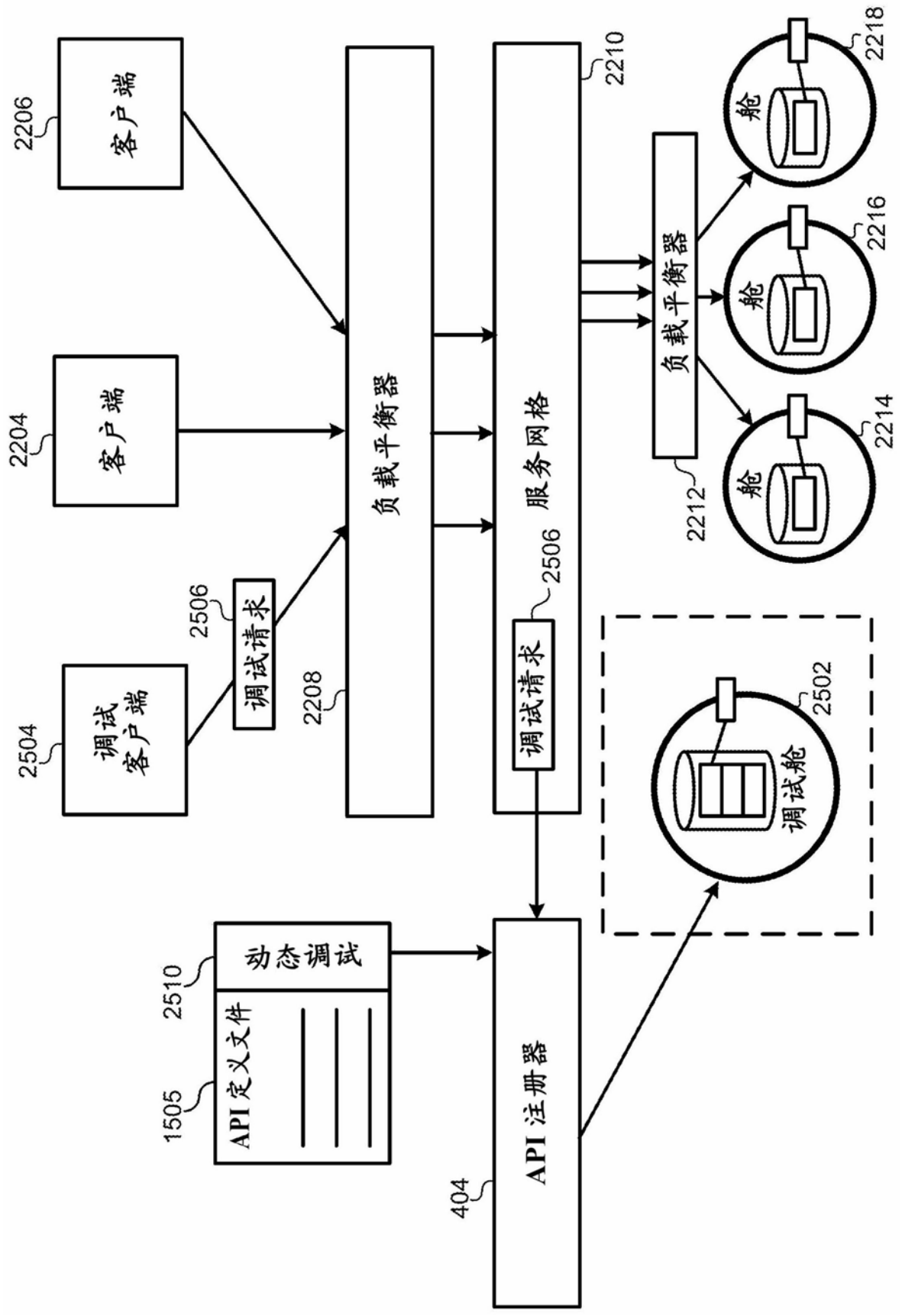

2206
客户端
2204
客户端
2504
调试客户端
2506
调试请求
2208
负载平衡器
2210
服务网格
2506
调试请求
2212
负载平衡器
2218
舱
2216
舱
2214
舱
2502
调试舱
2510
动态调试
1505
API 定义文件
404
API 注册器

图25

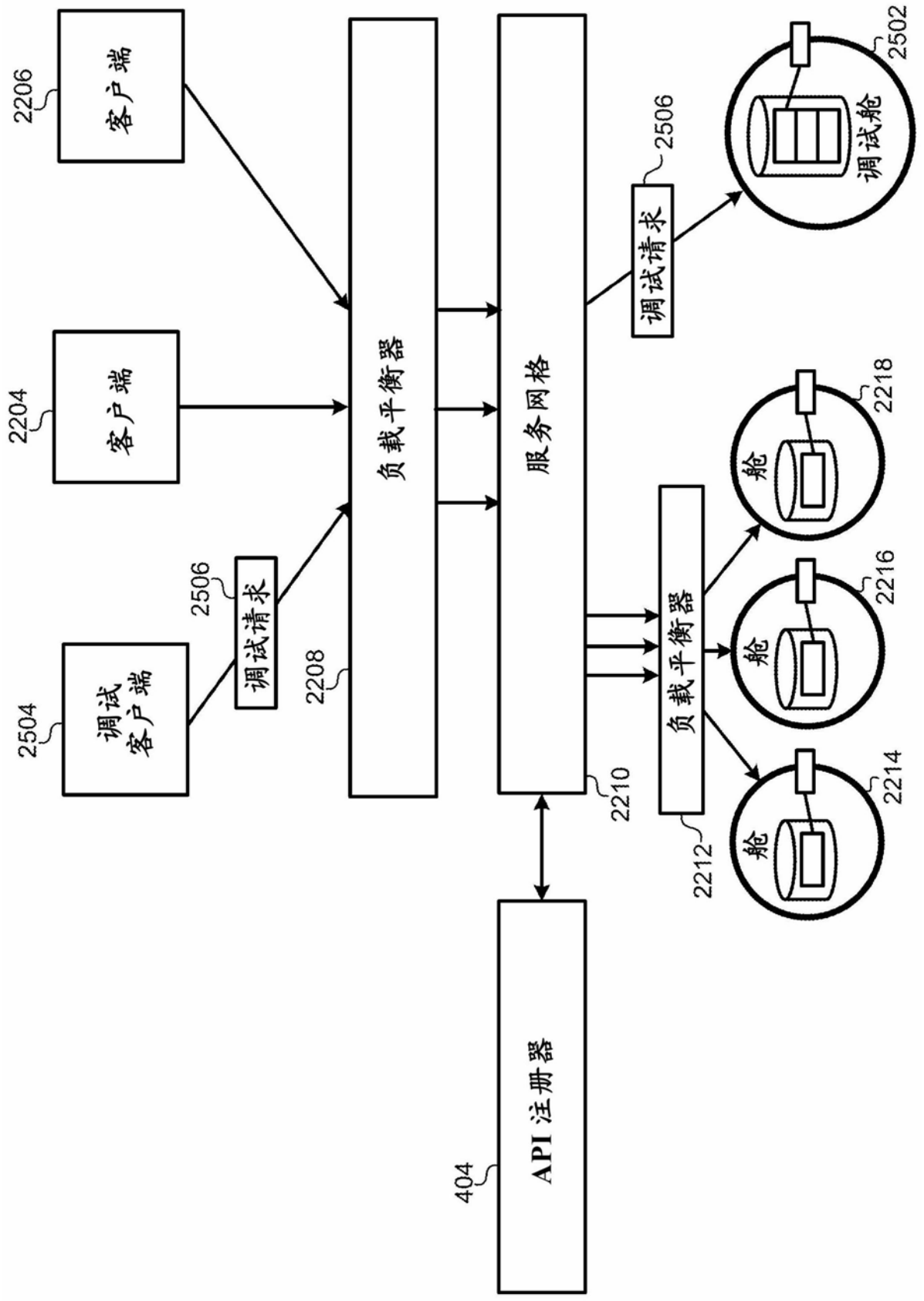

客户端
2206
客户端
2204
调试客户端
2504
2506
调试请求
2208
负载平衡器
服务网格
2210
2506
调试请求
2502
调试舱
2218
舱
2216
舱
2214
舱
负载平衡器
2212
API 注册器
404

图26

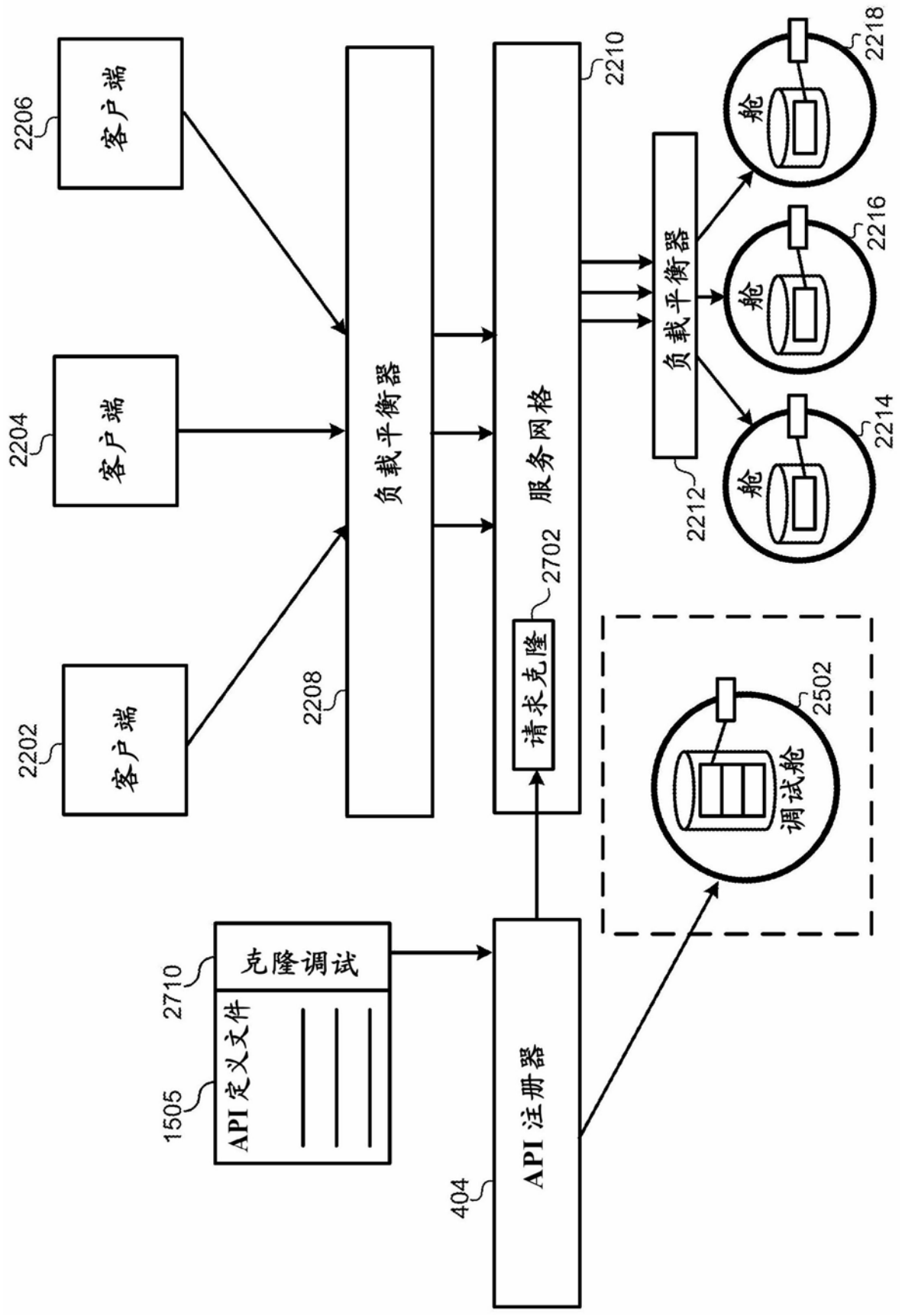

客户端
2206
客户端
2204
客户端
2202
负载平衡器
2208
服务网格
2210
请求克隆
2702
负载平衡器
2212
舱
2218
舱
2216
舱
2214
调试舱
2502
克隆调试
2710
API定义文件
1505
API 注册器
404

图27

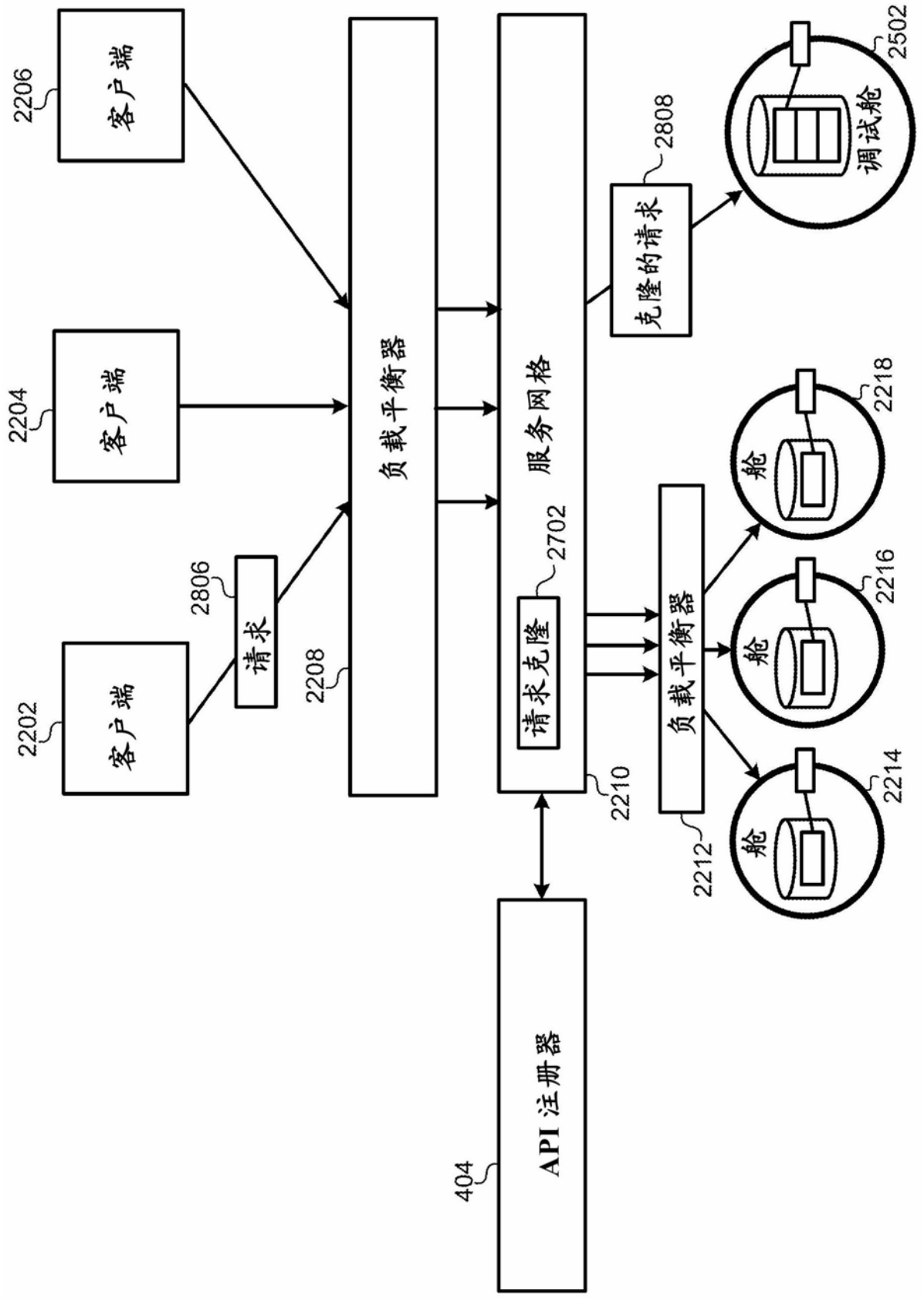

客户端
2206
客户端
2204
客户端
2202
2806
请求
2208
负载平衡器
服务网格
2702
请求克隆
2210
API 注册器
404
克隆的请求
2808
2502
调试舱
2218
舱
2216
舱
2214
舱
负载平衡器
2212

图28

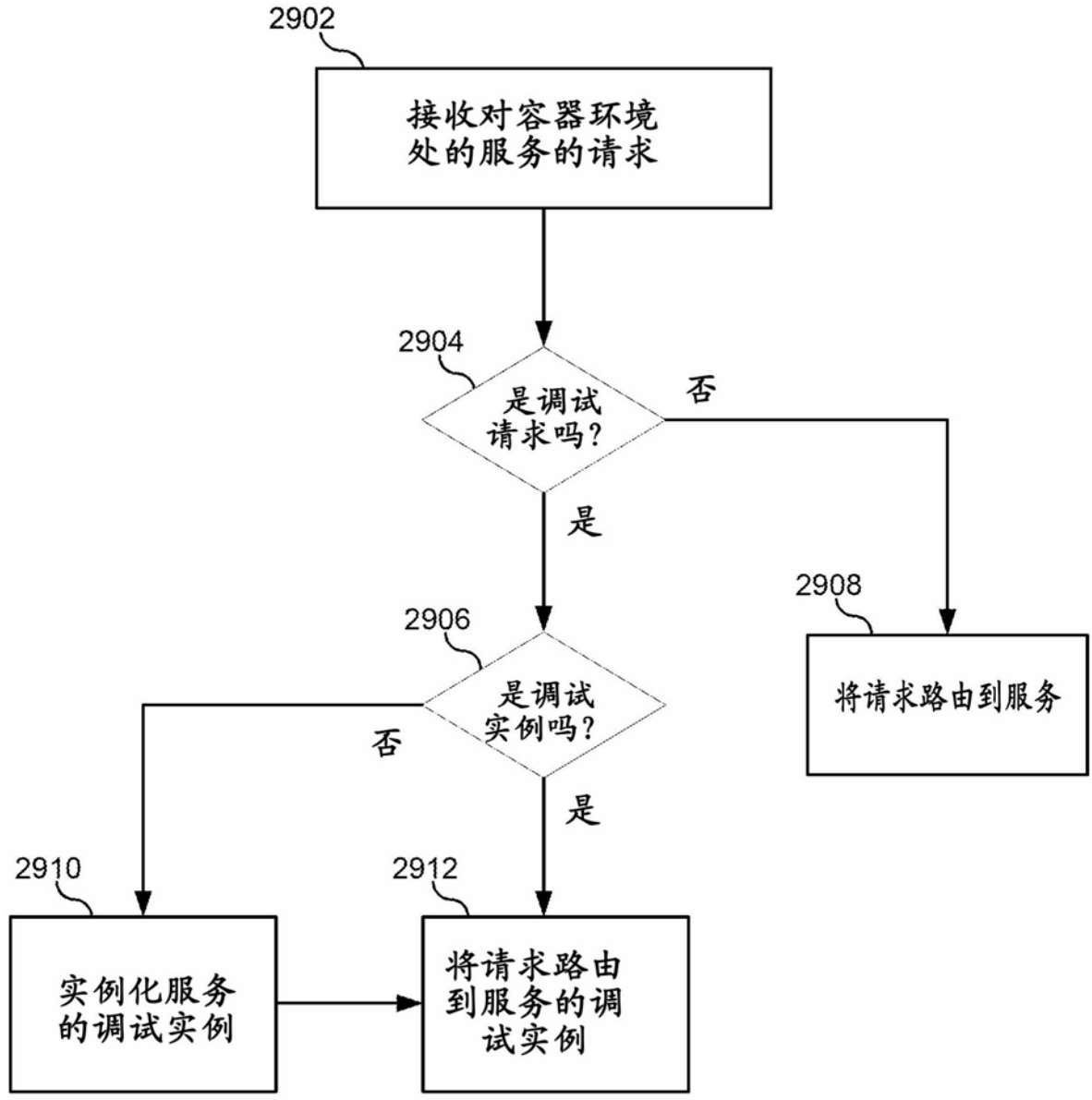

2902
接收对容器环境处的服务的请求
2904
是调试请求吗?
否
是
2908
将请求路由到服务
2906
是调试实例吗?
否
是
2910
实例化服务的调试实例
2912
将请求路由到服务的调试实例

图29

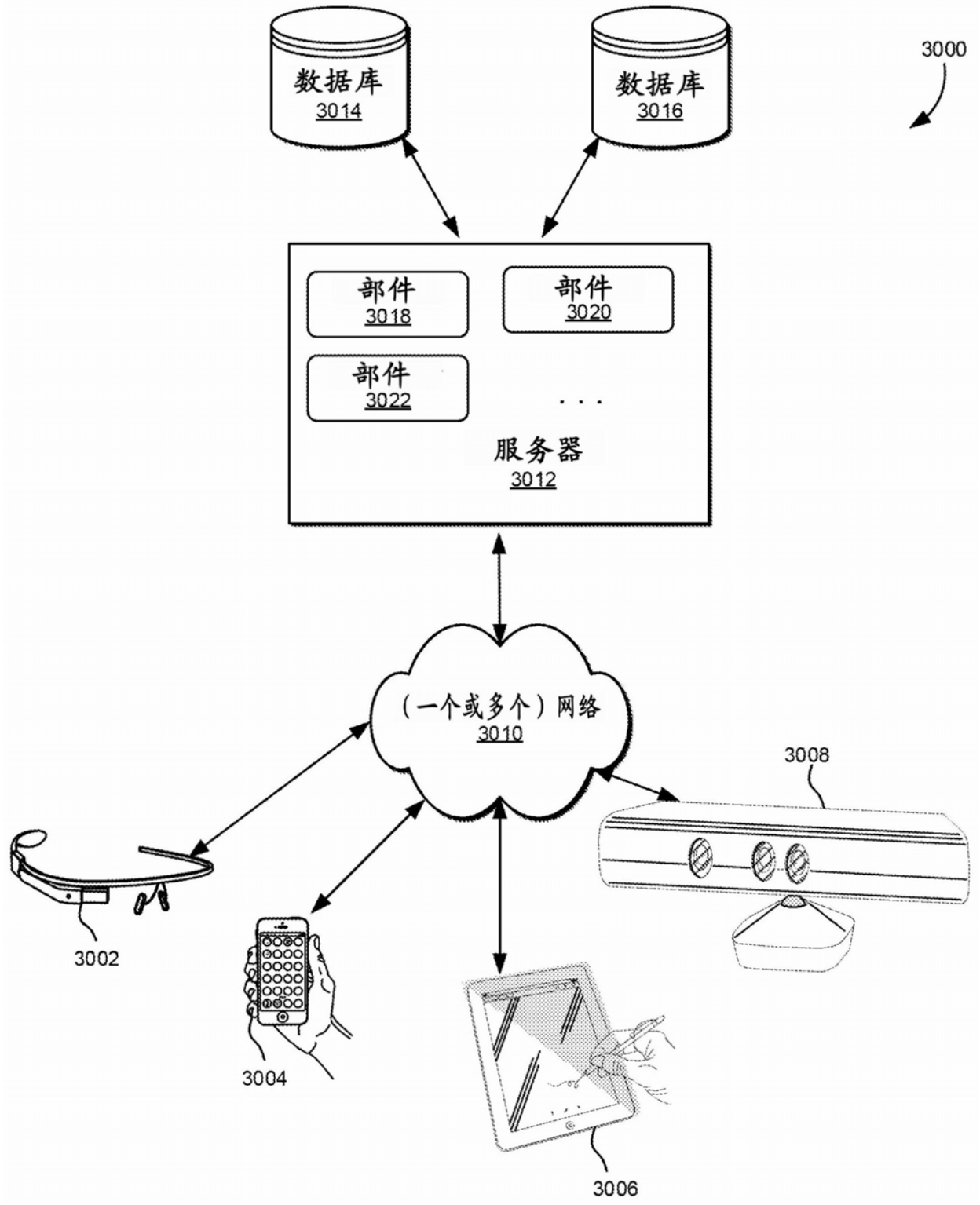

3000
数据库
3014
数据库
3016
部件
3018
部件
3020
部件
3022
. . .
服务器
3012
（一个或多个）网络
3010
3008
3002
3004
3006

图30

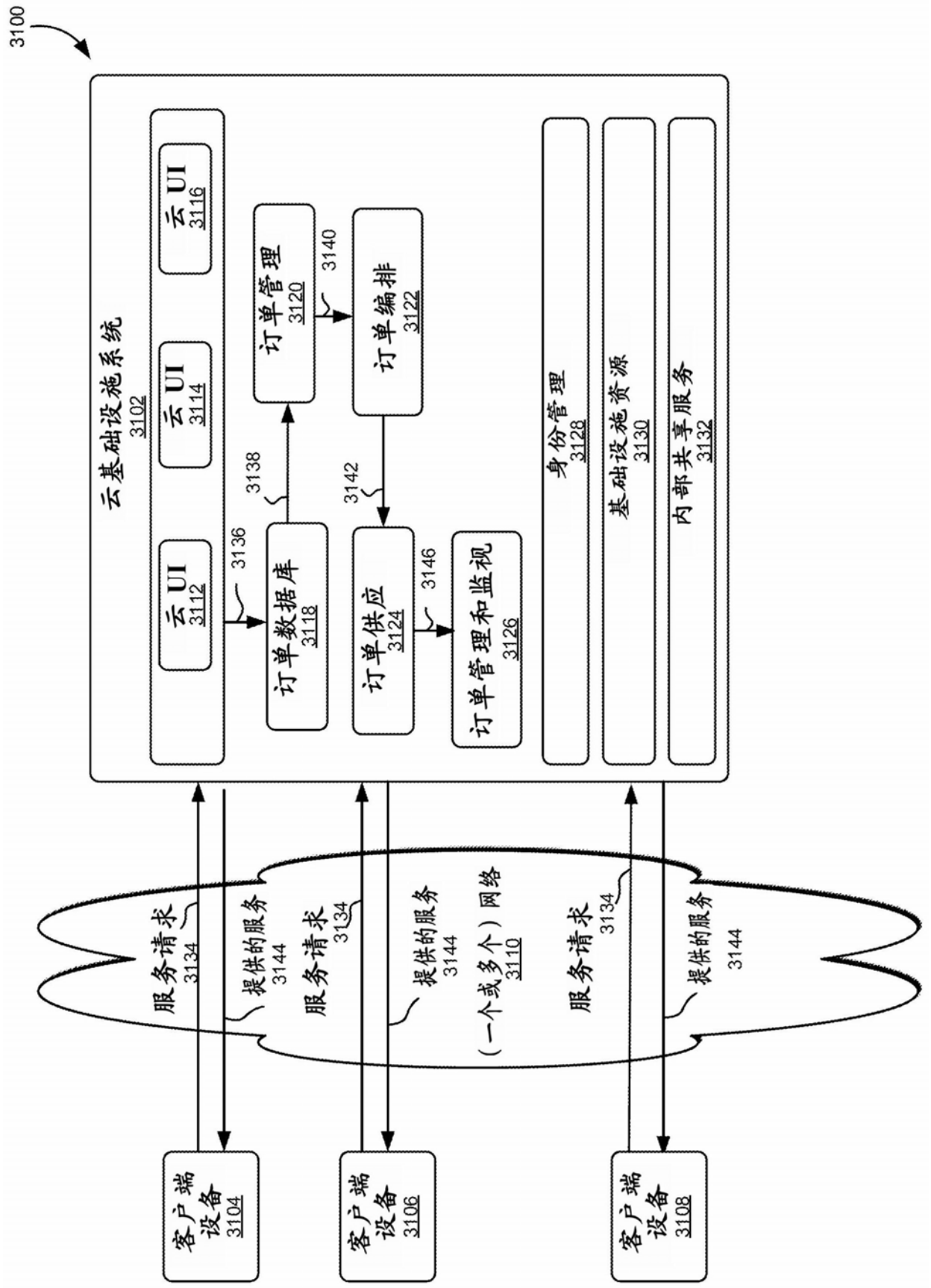

3100
云基础设施系统
3102
云 UI
3112
云 UI
3114
云 UI
3116
订单数据库
3118
订单管理
3120
订单编排
3122
订单供应
3124
订单管理和监视
3126
身份管理
3128
基础设施资源
3130
内部共享服务
3132
3136
3138
3140
3142
3146
服务请求
3134
提供的服务
3144
（一个或多个）网络
3110
客户端设备
3104
客户端设备
3106
客户端设备
3108

图31

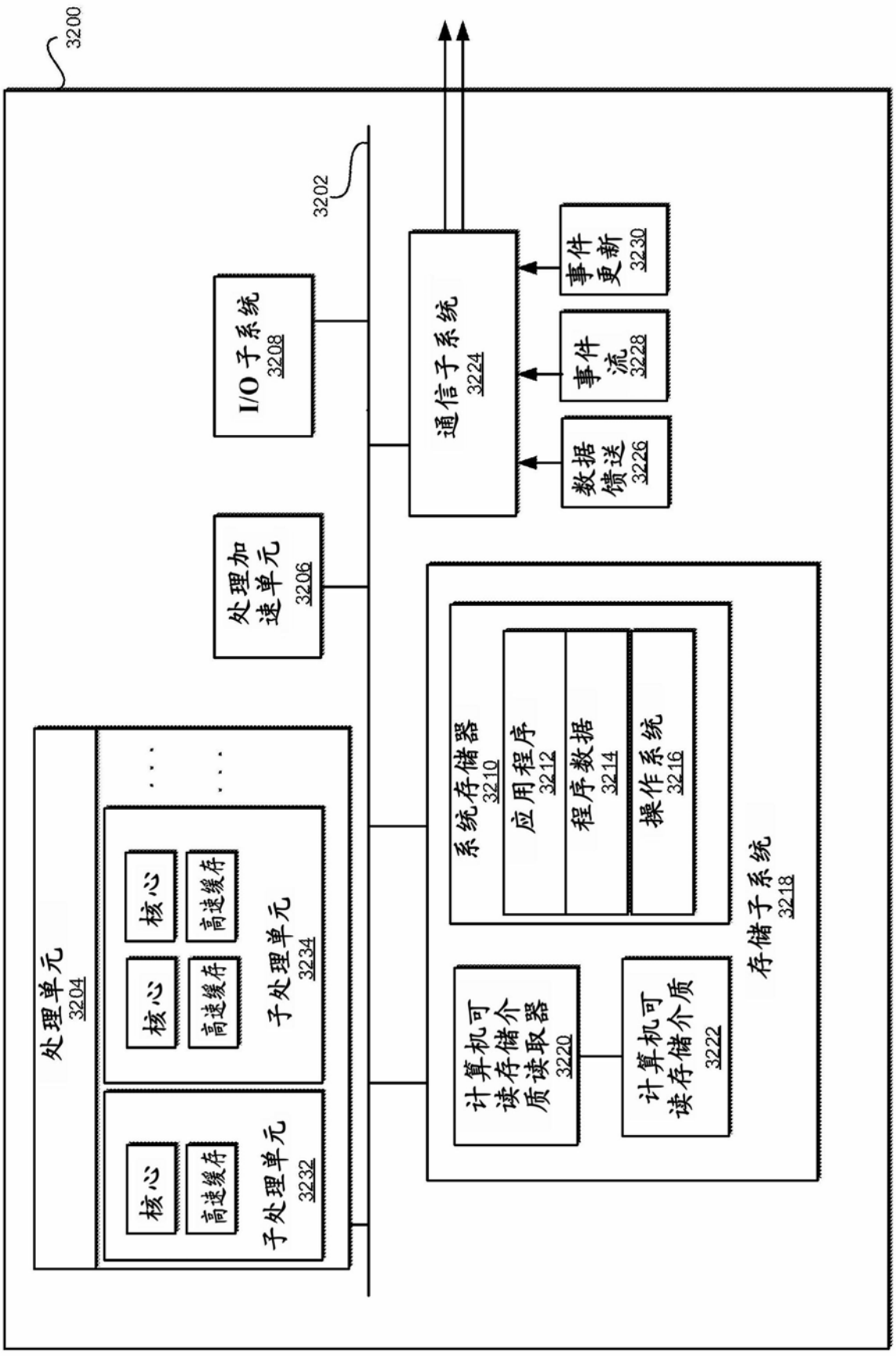

3200
3202
处理单元
3204
子处理单元
3232
核心
高速缓存
子处理单元
3234
核心
高速缓存
核心
高速缓存
I/O子系统
3208
处理加速单元
3206
通信子系统
3224
事件更新
3230
事件流
3228
数据馈送
3226
存储子系统
3218
系统存储器
3210
应用程序
3212
程序数据
3214
操作系统
3216
计算机可读存储介质读取器
3220
计算机可读存储介质
3222

图32