(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 110546615 A
(43)申请公布日 2019.12.06

(21)申请号 201880027451.0

(22)申请日 2018.04.27

(30)优先权数据
15/582,660 2017.04.29 US

(85)PCT国际申请进入国家阶段日
2019.10.25

(86)PCT国际申请的申请数据
PCT/US2018/029806 2018.04.27

(87)PCT国际申请的公布数据
WO2018/200961 EN 2018.11.01

(71)申请人 思科技术公司
地址 美国加利福尼亚州

(72)发明人 沃尔特·泰德·赫利克

(74)专利代理机构 北京东方亿思知识产权代理有限责任公司 11258
代理人 邓素敏

(51)Int.Cl.
G06F 9/54(2006.01)
G06F 11/34(2006.01)

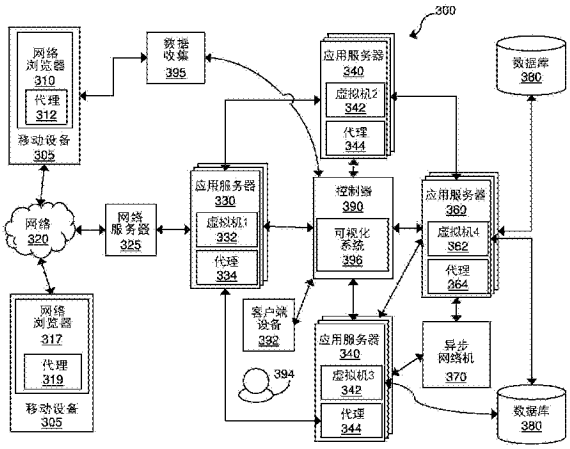
权利要求书3页 说明书14页 附图6页

(54)发明名称

超动态JAVA管理扩展

(57)摘要

一方面，公开了用于即时显露应用度量的系统。该系统包括：处理器；存储器；以及一个或多个模块，其存储在存储器中并可由处理器执行以执行各种操作。这些操作包括：初始化用于Java管理扩展(JMX)的Java托管对象；将经过初始化的Java托管对象附加到应用可访问的用于Java托管对象的资料库；在所附加的Java托管对象中创建对于属性字段的固定引用；在Java托管对象的初始化之后，向资料库提供所创建的固定引用；保存固定引用；描述将要创建的属性；以及在Java托管对象中创建新属性。



CN 110546615 A

1.一种用于即时显露应用度量的系统,该系统包括:

处理器;

存储器;以及

一个或多个模块,所述一个或多个模块被存储在所述存储器中并可由处理器执行以执行操作,所述操作包括:

初始化用于Java管理扩展(JMX)的Java托管对象;

将经初始化的Java托管对象附加到能被应用访问的、用于Java托管对象的资料库;

在所附加的Java托管对象中创建对于属性字段的固定引用;

在对所述Java托管对象的初始化之后,向所述资料库提供所创建的固定引用;

保存所述固定引用;

描述要创建的属性;以及

在所述Java托管对象中创建新属性,包括:

创建表示所述新属性的对象,

将表示所述新属性的所述对象放置到属性图中,

使用与所述新属性相关联的名称来引用所述属性图,

将来自所述属性图的信息注入与JMX兼容的属性信息对象阵列,

更新所述Java托管对象的所述属性字段以指向与JMX兼容的所述属性信息对象阵列。

2.如权利要求1所述的系统,其中,所述Java托管对象包括JMX托管Bean(MBean)。

3.如权利要求1或2所述的系统,其中,一个或多个模块被配置为使用反射链表达式来访问所述新属性。

4.如权利要求1至3中任一项所述的系统,其中,一个或多个模块被配置为修改所述新属性。

5.如权利要求4所述的系统,其中,一个或多个模块被配置为修改所述新属性,包括:

更新所述属性图;

将经更新的属性图转换到与JMX兼容的另一属性信息对象阵列;以及

更新所述Java托管对象中的所述属性字段以指向与JMX兼容的所述另一属性信息对象阵列。

6.如权利要求1至5中任一项所述的系统,其中,一个或多个模块被配置为在不重新启动Java虚拟机(JVM)的情况下即时创建所述新属性。

7.如权利要求1至6中任一项所述的系统,其中,一个或多个模块被配置为在不重新启动Java虚拟机(JVM)的情况下即时修改所述新属性。

8.如权利要求1至7中任一项所述的系统,其中,一个或多个模块被配置为使用反射链来即时创建并修改所述新属性。

9.一种用于即时显露应用度量的方法,所述方法包括:

初始化用于Java管理扩展(JMX)的Java托管对象;

将经初始化的Java托管对象附加到能被应用访问的、用于Java托管对象的资料库;

在所附加的Java托管对象中创建对于属性字段的固定引用;

在对所述Java托管对象的初始化之后,向所述资料库提供所创建的固定引用;

保存所述固定引用;

描述要创建的属性;以及

在所述Java托管对象中创建新属性,包括:

创建表示所述新属性的对象,

将表示所述新属性的所述对象放置到属性图中,

使用与所述新属性相关联的名称来引用所述属性图,

将来自所述属性图的信息注入与JMX兼容的属性信息对象阵列,

更新所述Java托管对象的所述属性字段以指向与JMX兼容的所述属性信息对象阵列。

10.如权利要求9所述的方法,其中,所述Java托管对象包括JMX托管Bean(MBean)。

11.如权利要求9或10所述的方法,包括:使用反射链表达式来访问所述新属性。

12.如权利要求9至11中任一项所述的方法,包括:修改所述新属性。

13.如权利要求12所述的方法,其中,修改所述新属性包括:

更新所述属性图;

将经更新的属性图转换为与JMX兼容的另一属性信息对象阵列;以及

更新所述Java托管对象中的所述属性字段以指向与JMX兼容的所述另一属性信息对象阵列。

14.如权利要求9至13中任一项所述的方法,其中,即时创建所述新属性是在不重新启动Java虚拟机(JVM)的情况下执行的。

15.如权利要求9至14中任一项所述的方法,其中,即时修改所述新属性是在不重新启动Java虚拟机(JVM)的情况下执行的。

16.如权利要求9至15中任一项所述的方法,其中,即时创建并修改所述新属性包括:使用反射链。

17.一种非暂态计算机可读介质,所述非暂态计算机可读介质包含指令,所述指令在被处理器执行时促使用于即时显露应用度量的操作被执行,所述操作包括:

初始化用于Java管理扩展(JMX)的Java托管对象;

将经初始化的Java托管对象附加到能被应用访问的、用于Java托管对象的资料库;

在所附加的Java托管对象中创建对于属性字段的固定引用;

在对所述Java托管对象的初始化之后,向所述资料库提供所创建的固定引用;

保存所述固定引用;

描述要创建的属性;以及

在所述Java托管对象中创建新属性,包括:

创建表示所述新属性的对象,

将表示所述新属性的所述对象放置到属性图中,

使用与所述新属性相关联的名称来引用所述属性图,

将来自所述属性图的信息注入与JMX兼容的属性信息对象阵列,

更新所述Java托管对象的所述属性字段以指向与JMX兼容的所述属性信息对象阵列。

18.如权利要求17所述的非暂态计算机可读介质,其中,所述Java托管对象包括JMX托管Bean(MBean)。

19.如权利要求17或18所述的非暂态计算机可读介质,其中,所述指令能被执行以使用反射链表达式来访问所述新属性。

20.如权利要求17至19中任一项所述的非暂态计算机可读介质，其中，所述指令能被访问以修改所述新属性。

21.一种用于即时显露应用度量的装置，所述装置包括：

用于初始化用于Java管理扩展(JMX)的Java托管对象的部件；

用于将经初始化的Java托管对象附加到能被应用访问的、用于Java托管对象的资料库的部件；

用于在所附加的Java托管对象中创建对于属性字段的固定引用的部件；

用于在对所述Java托管对象的初始化之后，向所述资料库提供所创建的固定引用的部件；

用于保存所述固定引用的部件；

用于描述要创建的属性的部件；以及

用于在所述Java托管对象中创建新属性的部件，包括：

用于创建表示所述新属性的对象的部件，

用于将表示所述新属性的所述对象放置到属性图中的部件，

用于使用与所述新属性相关联的名称来引用所述属性图的部件，

用于将来自所述属性图的信息注入与JMX兼容的属性信息对象阵列的部件，

用于更新所述Java托管对象的所述属性字段以指向与JMX兼容的所述属性信息对象阵列的部件。

22.如权利要求21所述的装置，还包括：用于实现根据权利要求10至16中任一项所述的方法的部件。

23.一种包括指令的计算机程序、计算机程序产品、或计算机可读介质，所述指令在被计算机执行时促使所述计算机执行根据权利要求9至16中任一项所述的方法的步骤。

超动态JAVA管理扩展

[0001] 相关申请的交叉引用

[0002] 本申请要求由Hulick于2017年4月29日递交的名为“HYPER DYNAMIC JAVA MANAGEMENT EXTENSION”的美国申请No.15/582,660的权益，其公开的内容通过引用结合于此。

背景技术

[0003] 自从Java5公布以来Java管理扩展(JMX)就已经存在，用于公布可以经由JMX查询在本地等级消费或者可以远程访问的度量的目的。JMX经由如下属性公布这些度量，该属性通过被称为MBean的托管Bean(Management Bean)显露出。MBean可以改变类型，但是被附加到诸如平台MBean服务器之类的MBean服务器。一旦被附加，它们就被显露给位于Java虚拟机(JVM)内部或外部并且可以经由若干远程协议选项访问的任意JMX客户端。

发明内容

[0004] 公开了用于被称为超动态JMX(Hyper Dynamic JMX)的新框架的实施方式的示例。具体地，所公开的用于被称为超动态JMX的新框架的技术被用来获取各种优点，这些优点包括对不需要代码改变的度量的动态(即，不需要重新启动)按需显露或者应用中的编码导致对于消费者的低TCO。所公开的用于新超动态JMX框架的技术可以被用来在运行时根据需要动态映射新商业度量或分析(其可以确实地访问任意类和任意方法，而不需要重新启动、重新编码等)。另外，所公开的技术可以被用于解决重大问题。

[0005] 独立权利要求中给出了本发明的多个方面，从属权利要求中给出了优选特征。一方面的特征可以单独应用于每个方面，或者可以与其他方面结合应用。

[0006] 一方面，公开了用于即时(on the fly)显露应用度量的系统。该系统包括处理器；存储器；以及一个或多个模块，其存储在存储器中并且可由处理器执行以执行各种操作。这些操作包括：初始化Java管理扩展(JMX)的Java托管对象(Java managed object)；将经过初始化的Java托管对象附加到应用可访问的用于Java托管对象的资料库(repository)；在所附加的Java托管对象中创建对于属性字段的固定引用；在Java托管对象的初始化之后，向资料库提供所创建的固定引用；保存固定引用；描述将要创建的属性；以及在Java托管对象中创建新属性。创建新属性包括：创建表示新属性的对象，将表示新属性的对象放置到属性图中，使用与新属性相关联的名称引用属性图，将来自属性图的信息注入与JMX兼容的属性信息对象阵列，以及更新Java托管对象的属性字段以指向与JMX兼容的属性信息对象阵列。

[0007] 可以通过各种方式将该系统实现为包括以下特征中的一个或多个特征。例如，Java托管对象可以包括JMX托管Bean(MBean)。一个或多个模块可以被配置为使用反射链表达式来访问(一个或多个)新属性。一个或多个模块可以被配置为修改(一个或多个)新属性。一个或多个模块可以被配置为修改新属性，包括：更新属性图；将经过更新的(一个或多个)属性的图转换为与JMX兼容的另一属性信息对象阵列；以及更新Java托管对象中的属性

字段以指向与JMX兼容的另一属性信息对象阵列。一个或多个模块可以被配置为在不重新启动Java虚拟机(JVM)的情况下即时创建(一个或多个)新属性。一个或多个模块可以被配置为在不重新启动Java虚拟机(JVM)的情况下即时修改(一个或多个)新属性。一个或多个模块可以被配置为使用反射链来即时创建并修改(一个或多个)新属性。

[0008] 另一方面,可以在用于即时显露应用度量的方法中实现所公开的技术。该方法包括:初始化Java管理扩展(JMX)的Java托管对象;将经过初始化的Java托管对象附加到应用可访问的用于Java托管对象的资料库;在所附加的Java托管对象中创建对于属性字段的固定引用;在Java托管对象的初始化之后,向资料库提供所创建的固定引用;保存固定引用;描述将要创建的属性;以及在Java托管对象中创建新属性。创建新属性包括:创建表示新属性的对象,将表示新属性的对象放置到属性图中,使用与新属性相关联的名称来引用属性图,将来自属性图的信息注入与JMX兼容的属性信息对象阵列,以及更新Java托管对象的属性字段以指向与JMX兼容的属性信息对象阵列。

[0009] 可以将该方法实现为包括以下特征中的一个或多个特征。例如,Java托管对象可以包括JMX托管Bean(MBean)。访问新属性包括使用反射链表达式。该方法可以包括修改新属性。修改新属性可以包括:更新属性图;将经过更新的属性图转换为与JMX兼容的另一属性信息对象阵列;以及更新Java托管对象中的属性字段以指向与JMX兼容的另一属性信息对象阵列。即时创建新属性可以在不重新启动Java虚拟机(JVM)的情况下执行。即时修改新属性可以在不重新启动Java虚拟机(JVM)的情况下执行。即时创建并修改新属性可以使用反射链执行。

[0010] 又一方面,可以在包括指令的非暂态计算机可读介质中实现所公开的技术,其中,所述指令在被处理器执行时促使用于即时显露应用度量的操作将被执行。所述操作包括:初始化Java管理扩展(JMX)的Java托管对象;将经过初始化的Java托管对象附加到应用可访问的用于Java托管对象的资料库;在所附加的Java托管对象中创建对于属性字段的固定引用;在Java托管对象的初始化之后,向资料库提供所创建的固定引用;保存固定引用;描述将创建的属性;以及在Java托管对象中创建新属性,包括:创建表示新属性的对象,将表示新属性的对象放置到属性图中,使用与新属性相关联的名称引用属性图,将来自属性图的信息注入与JMX兼容的属性信息对象阵列,以及更新Java托管对象的属性字段以指向与JMX兼容的属性信息对象阵列。

[0011] 可以将非暂态计算机可读介质实现为包括以下特征中的一个或多个特征。例如,Java托管对象包括JMX托管Bean(MBean)。所述指令可执行以使用反射链表达式来访问新属性。所述指令可访问以修改新属性。

附图说明

[0012] 图1A、图1B、图1C、图1D、和图1E是用于实现超动态JMX的示例性处理的处理流程图。

[0013] 图2是本专利文档所公开的可以实现超动态JMX的示例性应用智能平台的框图(包括针对图1A至图1E公开的处理)。

[0014] 图3是本专利文档所公开的用于实现超动态JMX的示例性系统的框图(包括针对图1A至图1E公开的处理)。

[0015] 图4是实现所公开的技术的示例性计算系统的框图。

具体实施方式

[0016] 互联网和万维网使得可用于几乎所有类型的商业的web服务激增。由于支持web服务的基础设施伴随的复杂性，越来越难跟随web服务的增多而保持最高水平的服务性能和用户体验。例如，横跨网络架构中的不同系统、工具、和层一起监控并记录数据提出了挑战。另外，即使在可以获取到数据时，也很难直接连接事件、原因、以及效果的链条。

[0017] 超动态JMX概述

[0018] JMX是用于显露将由本地或远程客户端使用的运行Java虚拟机(JVM)内的Java度量和操作的平台。JMX具有在由MBean平台服务器显露的MBean中的域、对象、和属性的名称空间下显露度量的接口。MBean是类似于JavaBean组件的托管Java对象，其遵循JMX规范中给出的设计模式。MBean可以表示设备、应用、或需要托管的任何资源。MXBean是仅引用预定义的数据类型集合的MBean类型。JMX连接器使得MBean服务器对于远程的基于Java技术的客户端是可访问的。

[0019] 操作允许执行MBean所显露的方法。属性定义MBean所显露的度量和度量数据类型。对象名称(ObjectName)是例如，域:key＝property的MBean名称。域是对象名称的前缀，通常指定“com.company.type...”。JConsole是广泛分布(利用Java)的JMX控制台(客户端)，用于浏览JMX度量并执行操作。反射是使用应用程序接口(API)检查、反思、并且在一些情况下改变程序运行时(例如，运行JVM)的行为的技术。反射链表达式是表示方法和/或字段的递归执行的字串，所述方法和/或字段以创建单个度量为意图产生实例值。一般，该链开始于提供起始实例的静态方法/字段或已知实例。

[0020] 最初，JMX MBean是显露度量的非常静态且必需的接口或实施方式。JMX MBean已经演变为动态型框架，以帮助经由配置文件甚至注解显露度量。但是，甚至动态JMX MBean也是有限的。例如，度量或属性在一般为可访问字段的一些数据结构中在运行时必需是预先确定的。一旦MBean被实例化，所有属性即被固定，而没有添加或移除的可能。另外，对于性质可为“虚拟”的数据/度量的即时瞄准是不可能的。

[0021] JMX支持不同的MBean类型。标准MBean使用简单的JavaBean型命名惯例和静态定义的管理接口。这是JBoss使用的最常见类型的MBean。动态MBean实现javax.management.DynamicMbean接口，并且在该组件被实例化用于最大灵活性时显露它们运行时的管理接口。在要被管理的组件在运行时间之前未知的情况下，JBoss利用动态MBean。开放MBean是动态MBean的扩展。开放MBean依赖用于通用可管理性的基本自描述、用户友好型数据类型。模型MBean也是动态MBean的扩展。模型MBean实现javax.management.modelmbean.ModelMBean接口。模型MBean通过提供默认行为来简化资源测试(instrumentation of resources)。

[0022] 本专利文件公开的技术提供了动态且高效的应用智能平台、系统、设备、方法、和包括非暂态类型在内的计算机可读介质，其中，该计算机可读介质包括用于促使包括处理器的机器执行本专利文档公开的各种操作以实现超动态JMX的指令。超动态JMX将动态MBean代理与反射链技术(有时称为取值器(getter)或赋值器(setter)链)组合在一起，以创建用于显露JMX属性(度量)的高动态的基础设施。

[0023] JMX动态MBean被模型MBean和开放MBean二者使用，并且表示显露运行时的属性和操作的能力(甚至在可能不知道它们由JMX显露的类中)。超动态MBean使用动态MBean接口，但是通过提供动态MBean中不存在的特征来增强动态方面。一方面，超动态MBean提供在JVM运行时即时显露并创建任意字段或度量的动态能力。例如，超动态MBean提供在MBean注册或实例化之后创建并移除属性和操作的能力。包括规则MBean、模型MBean、以及开放MBean在内的所有动态MBean被设计为仅在注册或实例化期间显露它们的属性和操作，并且这些属性和操作一旦被显露即变得不可改变。另外，即使字段和方法在该类外部是不可访问的，超动态MBean也提供使用反射链(方法/字段)即时动态访问字段和方法的能力。动态Mbean实施方式仅针对公共(可访问)取值/赋值方法，并且经由硬编码处理显露这些方法。

[0024] 超动态JMX技术和优点

[0025] 超动态JMX使用能够支持包括加法和减法的动态属性的单个MBean。超动态JMX使用属性到虚拟实施方式的动态映射，其中，该虚拟实施方式可以是固定字段或度量(不需要任何编码)。可以实现超动态JMX，以在两步骤处理中或即时解析并执行反射链表达式。反射链清理根单例模式(root singleton pattern)，例如对应用程序中任何位置的单例实例的静态引用以及任何其他根静态引用。这些技术允许超动态JMX提供各种优点。例如，超动态JMX可以即时显露并消耗消费者应用度量而不需要编码。超动态JMX可以无需重新启动JVM，在不进行编码的情况下即时显露并消耗JVM故障排除度量。

[0026] 超动态MBean模块

[0027] 可以使用超动态MBean模块实现所公开的用于超动态JMX的技术。图1A是示出使用超动态MBean模块实现超动态JMX的示例性处理100的处理流程图。超动态MBean模块可以在MBean平台服务器启动之后的任何时间，使用Java代理切换机制加载并启动。一旦被加载，超动态MBean模块即对其MBean进行初始化(102)，并将超动态MBean附加到平台MBean服务器(104)。平台MBean是用于监控封装部分JVM功能的JVM的MBean。平台MBean服务器是提供对于MBean的管理应用访问的MBean资料库。应用使用唯一对象名称通过平台MBean服务器访问MBean，而不直接访问MBean。超动态MBean模块在其MBean MBean信息(MBeanInfo)对象中创建对于属性字段的固定引用(106)并在初始化之后将该固定引用传递给平台MBean服务器(108)。保存该引用(110)。属性字段控制哪些属性是固定的。属性字段认为Mbean的属性是不可改变的。此时，根据配置创建属性并对属性进行初始化(112)或者简单地加载不具有属性的MBean(114)。

[0028] 图1B是使用本文档公开的超动态JMX技术来创建、使用、并修改属性的示例性处理120的处理流程图。描述将要创建的属性(122)。例如，可以针对将要创建的属性来描述名称、类型(java.lang.etc)、描述、以及虚拟位置(它们将是反射链表达式)。在MBean中创建并公布属性(124)。JMX客户端可以消耗所创建的JMX属性(126)。可以修改或移除所创建的属性(128)。

[0029] 图1C是用于在MBean中创建并公布属性的示例性处理124的处理流程图。用于在MBean中创建并公布属性的示例性处理124包括创建类似于JMX所使用的MBean属性信息(MBeanAttributeInfo)的对象(超动态MBean属性信息)(130)。在Mbean中创建并公布属性包括：处理反射链表达式并将其划分为运行时使用的多个步骤(132)。示例是：hd_hmxHD_JMX.serverFieldForTest.getClass().getClassLoader().getURSs().$length()。

[0030] 在MBean中创建并公布属性包括:将(用于属性的)对象放置于属性图中并通过其属性名称进行引用(134)。属性图表示超动态MBean内部属性。穿越属性图并将其注入JMX预期的内容,使用所保存的对于属性字段的固定引用,根据超动态MBean属性信息创建MBean属性信息对象阵列(136),其中,所保存的固定引用被更新为引用新属性阵列。

[0031] 经由JMX客户端操作,例如,JConsole,JMX属性被消耗。图1D是用于消耗JMX属性的示例性处理126的处理流程图。利用属性名称调用MBean的getAttribute或getAttributes方法(138)。MBean使用名称经由属性图查找属性,并提取包括经过处理的反射链表达式信息的超动态MBean属性信息(140)。执行反射链表达式并返回在JMX客户端中消耗的值或字符串(142)。

[0032] 图1E是用于移除或修改属性的示例性处理128的处理流示意图。更新属性图以移除或修改目标属性(144)。穿越经过更新的属性图(146)并将其转换为MBean属性信息对象阵列(148),然后更新属性字段以引用新属性阵列(150)。

[0033] 超动态JMX的示例性应用

[0034] 消费者可以选择度量浏览器选项,然后发起创建新JMX度量。响应于发起创建新JMX度量,可以向消费者呈现一种形式。消费者可以选择JMX需求信息或者键入JMX需求信息,以将具有属性名称的度量定义为该形式(form)。该形式还可以包括描述反射链(例如,使用向导助手)以对准将映射的真实度量的能力。一旦该形式完成(例如,由显示器上的按钮的按压指示),MBean接口类即被创建。然后,MBean impl(实施方式)类被创建。MBean实施方式属性取值器/设值器随后将调用通用映射,该通用映射将类或属性映射到将进行实际的反射工作并处理静态事物等的取值器/设值器反射链。这将利用MBean平台服务器进行注册。运行时,实施方式属性取值/设值方法将代理到取值器/设值器链,以提供作为度量的值。

[0035] 应用智能平台架构

[0036] 所公开的用于超动态JMX的技术可以用来增强用于应用性能管理的应用智能平台的性能。所公开的包括在不进行编码或重新启动JVM的情况下即时显露度量的能力的技术的很多优点可以提高用于应用性能管理的应用智能平台的性能。一方面,如下面针对图2至图4所讨论的,通过使用安装在实体处的单独机器处的代理对处于被监控环境中的诸如事务、层级、节点、以及机器之类的应用和实体进行监控,来检测与被监控环境有关的性能问题。例如,每个节点可以包括执行部分应用的一个或多个机器。代理收集与感兴趣的应用和正操作应用的相关节点和机器相关联的数据。所收集的数据的示例包括指示关系信息的诸如度量、元数据、以及拓扑数据之类的性能数据。代理收集的数据被提供给一个或多个控制器以分析该数据。

[0037] 图2是可以如本专利文档所公开地实现超动态JMX的示例性应用智能平台200的框图。该应用智能平台是监控并收集正被监控的应用环境的性能数据的度量的系统。在最简单的结构下,应用智能平台包括一个或多个代理210、212、214、216以及一个或多个控制器220。尽管图2示出了通信地链接到单个控制器的四个代理,但是代理和控制器的总数目可以基于包括被监控的应用的数目、应用环境如何分布、期望的监控等级、期望的用户体验等级等多个因素改变。

[0038] 控制器和代理

[0039] 控制器220是用于应用智能平台的中心处理和管理服务器。控制器220可以提供基于浏览器的用户界面(UI)230，该基于浏览器的用户界面(UI)230作为用于监控和分析被监控环境并排除被监控环境的故障的主要接口。控制器220可以控制并管理对分布在应用服务器上的商业事务的监控。具体地，控制器220可以从代理210、212、214、216以及协调器接收运行时数据，将商业事务数据的多个部分进行关联，与代理通信以配置运行时数据的收集，并通过界面230提供性能数据和报告。界面230可以被看作客户端设备240可查看的基于web的界面。在一些实施方式中，客户端设备240可以直接与控制器220通信，以查看用于监控数据的界面。

[0040] 在软件即服务(SaaS)实施方式中，控制器实例220由应用智能平台200的提供商远程托管。在预置(On-Prem)实施方式中，控制器实例220被本地安装并被自管理。

[0041] 控制器220可以从被部署以对被监控环境的最终(end)用户客户端、服务器、数据库和数据库服务器、以及应用进行监控的不同代理210、212、214、216接收数据。代理210、212、214、216中的任意代理可以实现为具有特定监控职责的不同类型的代理。例如，应用代理可以被安装在托管将要被监控的应用的每个服务器上。对代理进行测试向应用的运行时进程添加应用代理。

[0042] 数据库代理是安装在机器上的软件(例如，Java程序)，该机器对于被监控数据库和控制器具有网络入口。数据库代理向被监控以收集度量的数据库查询，并传递度量用于在度量浏览器中显示-并且在控制器UI的数据库页面中进行数据库监控。多个数据库代理可以向同一个控制器报告。附加的数据库代理可以被实现为备份数据库代理，以在故障或计划的机器停机时间期间接管主数据库代理。附加的数据库代理可以与主代理在相同或不同机器上运行。数据库代理可以被部署在被监控环境的每个不同网络中。多个数据库代理可以在相同机器上的不同用户账户下运行。

[0043] 独立的机器代理是从被监控环境中的服务器收集硬件相关性能统计数据的独立程序(例如，独立的Java程序)。独立的机器代理可以被部署在托管应用服务器、数据库服务器、管理服务器、web服务器等的机器上。独立的机器代理具有可扩展的架构。

[0044] 使用浏览器代理和移动代理执行最终用户监控(EUM)，以从客户端(例如，web浏览器或移动本机应用)的角度提供性能信息。浏览器代理和移动代理不同于通过服务器上的独立机器代理、数据库代理、应用代理的其他监控。通过EUM，可以根据监控需求监控web使用(例如，通过真实用户或合成代理)、移动使用、或任意组合。浏览器代理(例如，代理210、212、214、216)可以包括向控制器报告监控数据的报告器。

[0045] 浏览器代理是使用基于web的技术的小文件，例如，注入到每个被测试的web网页的JavaScript代理(尽可能接近顶级，当web网页被提供并收集数据时)。一旦web网页完成加载，所收集的数据即被捆绑到信标中并被发送到EUM云进行处理并准备好供控制器提取。浏览器真实用户监控(浏览器RUM)从真实或合成最终用户的角度提供对于web应用的性能的透视。例如，浏览器RUM可以确定特定的Ajax或iframe调用如何减慢页面加载时间以及服务器性能如何影响聚集或单独情况中的最终用户体验。

[0046] 移动代理是被添加到移动应用的源的小块的高性能代码。当最终用户实际使用移动应用时，移动RUM提供有关本地iOS或安卓移动应用的信息。移动RUM提供如下项的功能的可视性：移动应用本身、移动应用与所使用的网络的交互、以及与移动应用通信的任何服务

器侧应用。

[0047] 控制器220可以包括用于显示与所公开的技术有关的报告和仪表盘的可视化系统250。在一些实施例中，可视化系统250可以被实现在与托管控制器220的机器不同的单独机器(例如，服务器)中。

[0048] 应用智能监控

[0049] 所公开的技术可以通过监控应用环境提供应用智能数据，其中，该应用环境包括各种服务，诸如由应用服务器提供的web应用(例如，Java虚拟机(JVM)、互联网信息服务(IIS)、超文本预处理器(PHP)web服务器等)、数据库或其他数据存储区、以及诸如消息队列和高速缓存之类的远程服务。应用环境中的服务可以通过各种方式交互，以提供与应用的一组紧密结合的用户交互，例如，可用于最终用户消费者的一组用户服务。

[0050] 应用智能建模

[0051] 应用环境中的实体(例如，JBoss服务、MQSeries模块、和数据库)和由这些实体提供的服务(例如，登陆事务、服务或产品搜索、或购买事务)被映射到应用智能模型。在应用智能模型中，商业交易表示由被监控环境提供的特定服务。例如，在电子商务应用中，特定的真实世界服务可以包括用户登录、搜索物品、或向购物车添加物品。在内容入口中，特定的真实世界服务可以包括对于诸如，体育、商业、或娱乐新闻之类的内容的用户请求。在股票交易应用中，特定的真实世界服务可以包括诸如，接收股票报价、购买、或出售股票之类的操作。

[0052] 商业事务

[0053] 由被监控环境提供的特定服务的商业事务表示方式提供了关于参与处理特定请求的各种层级的上下文中的性能数据的视图。商业事务表示用于在被监控环境中实现服务请求的端到端处理路径。因此，商业环境是由入口点和横跨应用服务器、数据库、以及潜在的很多其他基础设施组件的处理路径定义的被监控环境中的一种类型的用户发起的动作。商业事务的每个实例是响应于特定用户请求而对该事务的执行。可以通过检测入口点的输入请求并横跨应用环境中的分布式组件跟踪与发起层处的请求相关联的活动，创建商业事务。可以针对商业事务生成流程图，该流程图示出了商业事务在应用环境中的接触点。

[0054] 性能监控可以由商业事务导向，以从最终用户的角度关注应用环境中的服务的性能。基于商业事务的性能监控可以提供关于服务是否可用(例如，用户可以登录、退出、或者查看其数据)、用户的响应时间、以及在出现问题时问题的原因方面的信息。

[0055] 商业应用

[0056] 商业应用是应用智能模型中的顶级容器(container)。商业应用包含相关服务和商业事务的集合。在一些实现方式中，可能需要单个商业应用来模拟该环境。在一些实施方式中，应用环境的应用智能模型可以被划分为多个商业应用。可以基于应用环境的细节，不同地组织商业应用。由于控制器UI中的基于角色的访问控件是由商业应用导向的，所以一个考虑是以反映特定组织中的工作团队的方式组织商业应用。

[0057] 节点

[0058] 应用智能模型中的节点对应于应用环境中的被监控的服务器或JVM。节点是模型化环境的最小单元。一般，节点对应于个体应用服务器、JVM、或CLR，其中，监控代理被安装在该应用服务器、JVM、或CLR上。每个节点在应用智能模型中识别其自身。节点处安装的代

理被配置为指定节点、层级、以及商业应用的名称，代理在该名称下向控制器报告数据。

[0059] 层级

[0060] 商业应用包含层级，应用智能模型中的单元包括一个或多个节点。每个节点表示经过测试的服务(例如，web应用)。尽管节点可以是应用环境中的不同应用，但是在应用智能模型中，节点是层级的元件，其与可能的很多其他层级组成总体逻辑商业应用。

[0061] 可以根据被监控应用环境的心智模型，在应用智能模型中组织层级。例如，可以将相同节点分在单个层级中(例如，冗余服务器的集群)。在一些实施方式中，可以出于将某些性能指标作为一个单元处理成单层级的目的，聚集任意(相同或不同)节点的集合。

[0062] 商业应用中的流量在层级中间流动，并且可以使用层级中间的连线在流程图中被可视化。另外，可以利用性能度量对指示层级之间的流量流的连线进行注解。在应用智能模型中，单个层级中的节点之间可以不存在任何互动。另外，在一些实施方式中，应用代理节点不能属于一个以上层级。类似地，机器代理不能属于一个以上层级。但是，一个以上机器代理可以被安装在机器上。

[0063] 后端系统

[0064] 后端是参与商业事务实例的处理的组件。后端不由代理测试。后端可以是web服务器、数据库、消息队列、或其他类型的服务。代理根据经过测试的代码辨认对于这些后端服务的调用(称为退出调用)。当服务没有被测试并且不能继续调用的事务上下文时，代理确定服务是后端组件。代理在后端响应时获取事务上下文，并且继续从这个位置开始遵循事务上下文。

[0065] 性能信息可以用于后端调用。对于由后端处理的事务的支柱的详细事务分析，需要测试数据库、web服务、或其他应用。

[0066] 基线和阈值

[0067] 应用智能平台使用自学习基线和可配置阈值来帮助识别应用问题。复杂的分布式应用具有大量性能度量，每个度量在一个或多个上下文中是重要的。在这种环境中，难以执行以下处理：确定对于特定度量正常的值或范围；设置所基于的有意义的阈值并接收相关警报；以及确定在应用或基础设施经受改变时什么是“正常”度量。出于这些原因，所公开的应用智能平台可以基于动态基线或阈值执行异常检测。

[0068] 所公开的应用智能平台自动计算所监控的度量的动态基线，其中，该动态基线基于实际使用情况定义对于每个度量什么是“正常的”。应用智能平台使用这些基线来识别其值掉出正常范围的后续度量。不再需要在迅速变化的应用环境中设置冗长的静态阈值(这容易出错)。

[0069] 所公开的应用智能平台可以使用可配置阈值来保持服务等级协议(SLA)，并通过检测慢、非常慢、以及安装的事务来确保系统的最佳性能等级。可配置阈值提供将正确的业务上下文与慢请求相关联的灵活方式，以隔离出根本原因。

[0070] 健康规则、策略、和动作

[0071] 另外，在性能问题正在出现或可能出现时，可以利用使用动态生成的基线触发警报或发起其他类型的修正动作的条件来设置健康规则。

[0072] 例如，动态基线可以被用来自动建立什么内容被当作特定应用的正常行为。可以相对于特定应用的基线或其他健康指示符来使用策略和健康规则，以在用户受到影响之前

检测并排除问题。健康规则可以被用来定义度量条件，以监控例如，何时“平均响应时间比基线慢四倍”。可以基于被监控应用环境来创建并修改健康规则。

[0073] 用于检验商业事务性能的健康规则的示例可以包括商业事务响应时间和商业事务误差率。例如，检验商业事务响应时间是否比正常情况高很多的健康规则可以定义临界条件为如下项的组合：比默认基线大3个标准偏差的平均响应时间和每分钟大于50次调用的负担。在一些实施方式中，这种健康规则可以定义报警条件为如下项的组合：比默认基线大2个标准偏差的平均响应时间和每分钟大于100次调用的负担。在一些实施方式中，检验商业事务误差率是否比正常情况高很多的健康规则可以定义临界条件为如下项的组合：比默认基线大3个标准偏差的误差率、每分钟大于10个误差的误差率、以及每分钟大于50次调用的负担。在一些实施方式中，这种健康规则可以定义报警条件为如下项的组合：比默认基线大2个标准偏差的误差率、每分钟大于5个误差的误差率、以及每分钟大于50次调用的负担。这些是健康规则的非穷尽且非限制性的示例，并且可以根据用户需要定义其他健康规则。

[0074] 策略可以被配置为在违反健康规则时或者在任何事件发生时触发动作。所触发的动作可以包括通知、诊断动作、自动缩放能力、运行修正脚本。

[0075] 度量

[0076] 大多数度量与应用服务器基础架构的总体性能(例如，CPU繁忙百分比、被使用的存储器的百分比等)或应用或商业事务的总体性能(例如，负担、平均响应时间、误差率等)有关。控制器UI中的度量浏览器可以被用来查看代理报告给控制器的所有度量。

[0077] 另外，被称为信息点的特定度量可以被创建，以报告给定商业(相对于给定应用)如何执行。例如，某个产品或产品集的总收益的性能可以被监控。另外，信息点可以被用于报告如何执行给定代码，例如，特定方法被调用几次以及执行花费多少时间。另外，使用机器代理的扩展可以被创建，以报告用户定义的自定义度量。这些自定义度量是基本的并且在控制器中被报告，就像内置度量一样。

[0078] 可以使用表述性状态转移(REST)API以编程方式访问所有度量，其中，该REST API返回JavaScript对象符号(JSON)或可扩展标记语言(XML)格式。另外，REST API可以被用来查询并操纵应用环境。

[0079] 快照

[0080] 快照提供给定应用在某个时间点的详细图片。快照通常包括调用图，该调用图允许下挖到可能导致性能问题的代码行。最常见的快照是事务快照。

[0081] 应用智能平台的示例性实施方式

[0082] 图3是本专利文档公开的用于执行超动态JMX的示例性系统300的框图(包括参考图1A、图1B、图1C、图1D、和图1E公开的处理)。图3的系统300包括客户端设备305和392、移动设备315、网络320、网络服务器325、应用服务器330、340、350、和360、异步网络机370、数据存储区380和385、控制器390、以及数据收集服务器395。控制器390可以包括用于提供针对超动态JMX生成的报告的显示的可视化系统396，如本专利文档所公开的。在一些实施方式中，可视化系统396可以被实现在不同于托管控制器390的机器的单个机器(例如，服务器)中。

[0083] 客户端设备305可以包括网络浏览器310，并且可以被实现为例如，膝上型计算设

备、桌面型计算设备、工作台、或一些其他计算设备之类的计算设备。网络浏览器310可以是用于在网络320上经由网络服务器325查看诸如应用服务器330之类的应用服务器提供的内容的客户端应用。

[0084] 网络浏览器310可以包括代理312。当网络浏览器向服务器添加、下载、或者以其他方式加载应用时，代理312可以被安装在网络浏览器310和/或客户端305。代理312可以被执行，以监控网络浏览器310、客户端305的操作系统、以及客户端305的任何其他应用、API、或另一组件。代理312可以确定网络浏览器导航时间度量，访问浏览器cookie，监控代码，并且向数据收集360、控制器390、或另一设备发送数据。代理312可以执行与监控客户端305处的请求或网络有关的其他操作(如本文所讨论的，包括报告生成)。

[0085] 移动设备315连接到网络320，并且可以被实现为适用于在网络上发送和接收内容的便携设备(例如，移动电话、智能电话、平板计算机、或其他便携设备)。客户端设备305和移动设备315二者都可以包括被配置为访问网络服务器325提供的web服务的硬件和/或软件。

[0086] 移动设备315可以包括网络浏览器317和代理319。移动设备还可以包括客户端应用和可以由代理319监控的其他代码。代理319可以驻留在网络浏览器317中和/或与网络浏览器317通信，并且与其他应用、操作系统、API、以及移动设备315上的其他硬件和软件通信。代理319可以具有与针对客户端305上的代理312所描述的类似功能，并且可以向数据收集服务器360和/或控制器390报告数据。

[0087] 网络320可以帮助系统300的不同服务器、设备、以及机器中间的数据通信(一些连接利用去往网络320的线条示出，一些未示出)。网络可以被实现为专用网、公用网、内联网、互联网、蜂窝网、Wi-Fi网、VoIP网、或者一个或多个这些网络的组合。网络320可以包括诸如负载均衡机器和其他机器之类的一个或多个机器。

[0088] 网络服务器325连接到网络320，并且可以接收并处理在网络320上接收的请求。网络服务器325可以被实现为实现网络服务的一个或多个服务器，并且可以与应用服务器330实现在相同的机器上或者实现在一个或多个不同机器上。当网络320为互联网时，网络服务器325可以被实现为web服务器。

[0089] 应用服务器330与网络服务器325、应用服务器340和350、以及控制器390通信。应用服务器350还可以与其他机器和设备(图3未示出)通信。应用服务器330可以托管应用或分布式应用的各部分。主机应用332可以在例如，包括Java、PHP、.Net、以及Node.JS在内的很多平台中的一个平台中，可以被实现为Java虚拟机，或者可以包括一些其他主机类型。应用服务器330还可以包括一个或多个代理334(即，“模块”)，该一个或多个代理包括语言代理、机器代理、网络代理、以及其他软件模块。应用服务器330可以被实现为图3所示的一个或多个服务器。

[0090] 可以使用字节代码插入或字节代码测试(BCI)对应用332和应用服务器330上的其他软件进行测试，以修改该应用或其他软件的对象代码。经过测试的对象代码可以包括用于如下操作的代码：检测应用332接收到的调用、应用332发送的调用，并且在该应用的执行期间与代理334通信。BCI还可以被用来监控应用和/或应用服务器的一个或多个套接字，以监控套接字并捕捉在该套接字上到来的分组。

[0091] 在一些实施例中，服务器330可以包括应用和/或代码而不包括虚拟机。例如，服务

器330、340、350、和360均可以包括Java代码、.Net代码、PHP代码、Ruby代码、C代码、C++代码、或其他二进制代码，以实现应用并处理从远程源接收到的请求。相对于应用服务器对虚拟机的引用仅用于示例性目的。

[0092] 应用服务器330上的代理334可以被安装、下载、嵌入、或以其他方式提供在应用服务器330上。例如，代理334可以通过测试对象代码、向服务器下载代理、或者一些其他方式被提供在服务器330中。代理334可以被执行以监控应用服务器330，监控虚拟机332中运行的代码(或其他程序语言，例如，PHP、.Net、或C程序)、机器资源、网络层数据，并且与应用服务器330上的字节测试代码和应用服务器330上的一个或多个应用通信。

[0093] 代理334、344、354、和364中的每个代理可以包括诸如语言代理、机器代理、和网络代理之类的一个或多个代理。语言代理可以是适合在特定主机上运行的代理类型。语言代理的示例包括JAVA代理、.Net代理、PHP代理、以及其他代理。机器代理可以从其所安装在的特定机器收集数据。网络代理可以捕捉网络信息，例如，从套接字收集的数据。

[0094] 代理334可以检测诸如应用服务器330接收调用和发送请求的操作、资源使用、以及输入分组。代理334可以接收数据，通过例如将数据聚集到度量中来处理数据，并且向控制器390发送数据和/或度量。代理334可以执行与监控本文讨论的应用和应用服务器330有关的其他操作。例如，代理334可以识别其他应用，共享商业事务数据，聚集检测到的运行时间数据，以及进行其他操作。

[0095] 代理可以操作以监控节点、层级、节点、或其他实体。节点可以是软件程序或硬件组件(例如，存储器、处理器等)。节点的层级可以包括可以处理类似商业事务的多个节点，节点的层级可以位于相同的服务器上，并且可以以其他方式相互关联，或者可以不相互关联。

[0096] 语言代理可以是适用于测试或修改、收集来自主机的数据并驻留在主机上的代理。主机可以是Java、PHP、.Net、Node.JS、或其他类型的平台。语言代理可以收集流数据以及与特定应用的执行相关联的数据。语言代理可以测试应用的最低等级，以收集流数据。流数据可以指示哪个层级正在与哪个层级通信以及在哪个端口上通信。在一些实例中，从语言代理收集的流数据包括源IP、源端口、目的地IP、以及目的地端口。语言代理可以向控制器报告应用数据和调用链数据。语言代理可以向网络代理报告与特定应用相关联的所收集的流数据。

[0097] 网络代理可以是驻留在主机上并收集网络流群组数据的单独代理。网络流群组数据可以包括由网络代理所安装在的应用接收到的网络流的协议信息、源IP、目的地端口、以及目的地IP。网络代理可以通过拦截来自一个或多个套接字的分组并对这些分组执行分组捕捉来收集数据。网络代理可以从与要监控的应用相关联的语言代理接收流数据。对于匹配语言代理所提供的流数据的流群组中的流，网络代理汇总流数据以确定诸如，TCP吞吐量、TCP损失、延时、和带宽之类的度量。网络代理随后可以向控制器报告度量、流群组数据、以及调用链数据。网络代理还可以在应用服务器处进行系统调用，以确定诸如，主机状态检测、网络状态检测、套接字状态、和其他信息之类的系统信息。

[0098] 机器代理可以驻留在主机上，并且收集关于实现该主机的机器的信息。机器代理可以收集诸如处理器使用、存储器使用、以及其他硬件信息之类的信息并从该信息生成度量。

[0099] 语言代理、网络代理、以及机器代理中的每个代理可以向控制器报告数据。控制器390可以被实现为与位于一个或多个服务器或机器上的代理通信的远程服务器。在由一个或多个被监控应用实现并且存在于一个或多个被监控网络上的分布式应用的背景中，控制器可以接收度量、调用链路数据和其他数据、关联作为分布式事务的部分的接收数据、并且报告关联数据。控制器可以为用户提供报告、一个或多个用户界面、以及其他信息。

[0100] 代理334可以针对服务器330接收的请求(例如，由与用户或其他源相关联的客户端305或315接收的请求)创建请求标识符。请求标识符可以被发送到客户端305或移动设备315中发送该请求的设备。在实施例中，当数据被收集并分析用于特定商业事务时，可以创建请求标识符。

[0101] 应用服务器340、350、和360中的每个应用服务器可以包括应用和代理。每个应用可以在相应的应用服务器上运行。应用服务器340-360上的应用342、352、和362中的每个应用可以类似于应用332操作，并且执行分布式商业事务的至少一部分。代理344、354、和364可以监控应用342-362，收集并处理运行时的数据，并与控制器390通信。应用332、342、352、和362可以相互通信，作为执行分布式事务的一部分。每个应用可以调用另一虚拟机的任意应用或方法。

[0102] 异步网络机370可以参加与诸如应用服务器350和360之类的一个或多个应用服务器的异步通信。例如，应用服务器350可以向异步网络机发送多个呼叫或消息。不是传送回应用服务器350，异步网络机可以处理消息，并最终向应用服务器360提供响应(例如，经过处理的消息)。由于不存在从异步网络机到应用服务器360的返回消息，所以它们之间的通信是异步的。

[0103] 数据存储区380和385都可以由诸如应用服务器350之类的应用服务器访问。数据存储区385也可以由应用服务器350访问。数据存储区380和385中的每个数据存储区可以存储数据、处理数据、并且返回从应用服务器接收到的查询。数据存储区380和385中的每个数据存储区可以包括或不包括代理。

[0104] 控制器390可以控制并管理对分布在应用服务器330-360上的商业事务的监控。在一些实施例中，控制器390可以从数据收集服务器360接收包括与监控客户端305和移动设备315处的客户端请求相关联的数据在内的应用数据。在一些实施例中，控制器390可以从代理312、319、334、344、和354中的每个代理接收应用监控数据和网络数据。控制器390可以对商业事务数据的多个部分进行关联，与代理通信以配置数据的收集，并通过界面提供性能数据和报告。该界面可以被看作客户端设备392可查看的基于web的界面，客户端设备392可以是移动设备、客户端设备、或者用于查看控制器390所提供的界面的任何其他平台。在一些实施例中，客户端设备392可以直接与控制器390通信，以查看用于监控数据的界面。

[0105] 客户端设备392可以包括任意计算设备，该任意计算设备包括移动设备或客户端计算机(例如，桌面型计算设备、工作台、或其他计算设备)。客户端计算机392可以与控制器390通信，以创建并查看自定义界面。在一些实施例中，控制器390提供用于创建并查看作为内容页(例如，web网页)的自定义界面的界面，其中，该内容页可以被提供给客户端设备392上的网络浏览器应用并通过该网络浏览器应用被呈现。

[0106] 应用332、342、352、和362可以是多种类型的应用中的任意应用。可以实现应用332-362的应用示例包括Java、PHP、.Net、Node.JS、以及其他应用。

[0107] 图4是用于实现本技术的计算机系统400的框图。图4的系统400可以被实现在客户端405、492，网络服务器425，服务器430、440、450、460，异步网络机470、以及控制器490的背景中。

[0108] 图4的计算系统400包括一个或多个处理器410和存储器420。主存储器420部分地存储供处理器410执行的数据和指令。主存储器410可以存储运行时的可执行代码。图4的系统400进一步包括大容量存储设备430、(一个或多个)便携存储介质驱动440、输出设备450、用户输入设备460、图形显示器470、以及外围设备480。

[0109] 图4所示的组件被描述为经由单个总线490连接。但是，组件可以通过一个或多个数据传输机构连接。例如，处理器单元410和主存储器420可以经由本地微处理器总线连接，并且大容量存储设备430、(一个或多个)外围设备480、便携或远程存储设备440、以及显示系统470可以经由一个或多个输入/输出(I/O)总线连接。

[0110] 可以利用磁盘驱动或光盘驱动实现的大容量存储设备430是用于存储供处理器单元410使用的数据和指令的非易失性存储设备。大容量存储设备430可以存储用于实现本发明的实施例的系统软件，以用于将该软件加载到主存储器420中的目的。

[0111] 便携存储设备440结合诸如压缩盘、数字视频盘、磁盘、闪存之类的便携非易失性存储介质操作，以输入和输出去往和来自图4的计算机系统400的数据和代码。用于实现本发明的实施例的系统软件可以被存储在这种便携介质上，并且可以经由便携存储设备440被输入到计算机系统400。

[0112] 输入设备460提供用户接口的一部分。输入设备460可以包括用于输入字母数字和其他信息的诸如键盘之类的字母-数字键盘，或诸如鼠标、跟踪球、指示笔、或光标方向键之类的指针设备。另外，图4所示的系统400包括输出设备450。适当输出设备的示例包括扬声器、印刷器、网络接口、以及监视器。

[0113] 显示系统470可以包括液晶显示器(LCD)或其他适当显示设备。显示系统470接收文本和图形信息，并且处理该信息用于输出到显示设备。

[0114] 外围设备480可以包括任何类型的计算机支持设备，以向计算机系统添加附加功能。例如，(一个或多个)外围设备480可以包括调制解调器或路由器。

[0115] 图4的计算机系统400中包含的组件可以包括个人计算机、手持计算设备、电话、移动计算设备、工作台、服务器、迷你计算机、大型计算机、或任何其他计算设备。计算机还可以包括不同的总线配置、联网平台、多处理器平台等。可以使用包括Unix、Linux、Windows、Apple OS、以及其他适当操作系统的各种操作系统(包括移动版本)。

[0116] 当实现诸如智能电话或平板计算机的移动设备时，图4的计算机系统400可以包括一个或多个天线、无线电设备、以及用于在无线信号上进行通信(例如，使用Wi-Fi、蜂窝、或其他无线信号的通信)的其他电路。

[0117] 尽管本专利文档包含很多细节，但是这些细节不应该被理解为对任何发明或请求保护的范围的限制，而应该被理解为特定发明的特定实施例特有的特征的描述。在本专利文档中在不同实施例的背景中描述的某些特征也可以结合单个实施例实现。相反，在单个实施例的背景中描述的各种特征也可以分别实现在多个实施例中或者实现在任何适当的子组合中。另外，尽管以上将特征描述为在某些组合甚至如最初请求保护地那样动作，但是来自请求保护的组合中的一个或多个特征在一些情况中可以被从该组合删除，并且请求保

护的组合涉及子组合或子组合的变形。

[0118] 类似地，当在附图中以特定顺序描绘操作时，这不应该被理解为这些操作必须以所示出的特定顺序或相继次序被执行，或者所有示出的操作都必须被执行，以实现期望结果。另外，本专利文档中描述的实施例的各种系统组件的分割不应该被理解为在所有实施例中都需要这种分割。

[0119] 仅描述了少量实施方式和示例，并且可以基于本专利文档中描述和示出的内容作出其他实施方式、增强、和变化。

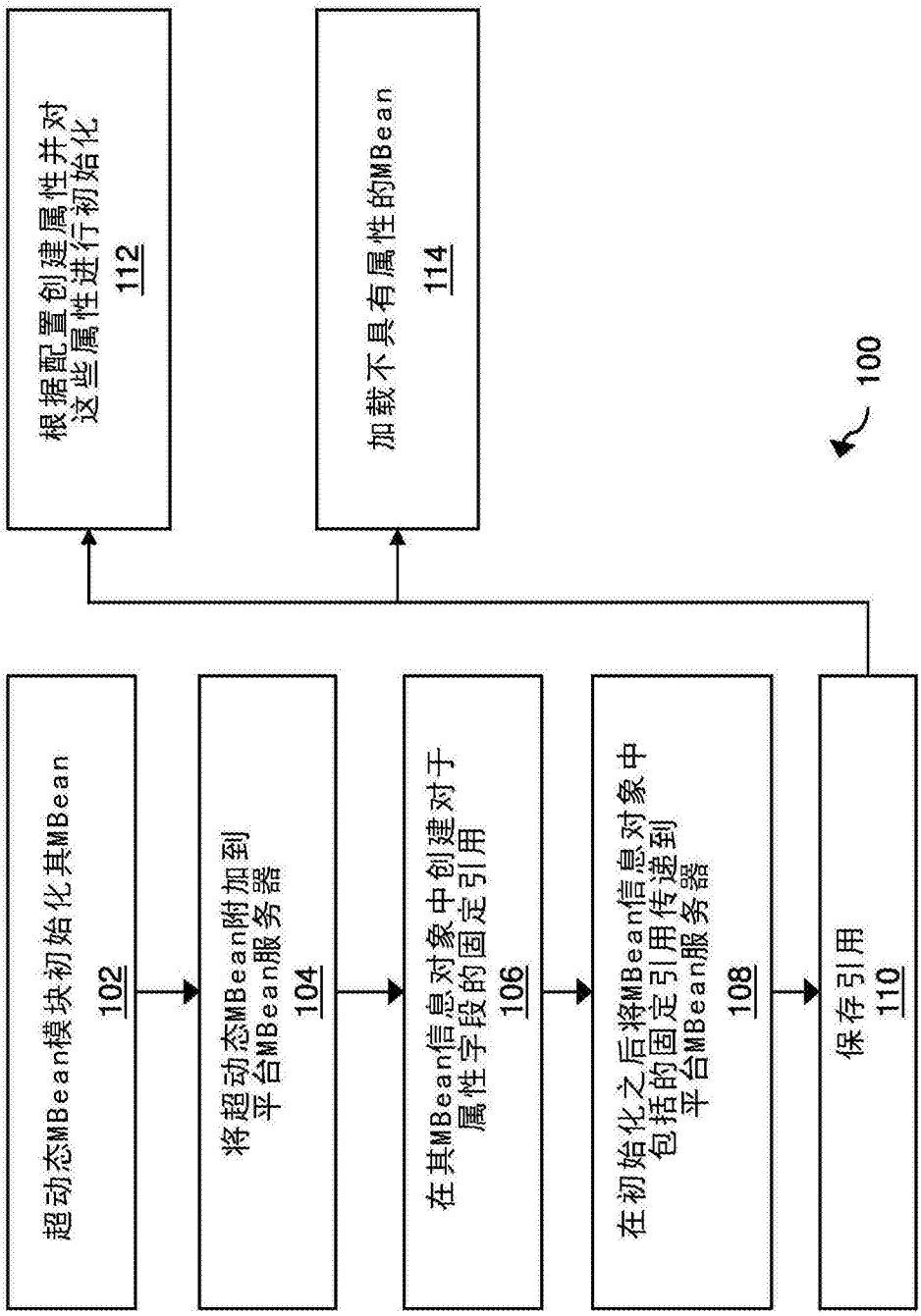

超动态MBean模块初始化其MBean
102
将超动态MBean附加到
平台MBean服务器
104
在其MBean信息对象中创建对于
属性字段的固定引用
106
在初始化之后将MBean信息对象中
包括的固定引用传递到
平台MBean服务器
108
保存引用
110
根据配置创建属性并对
这些属性进行初始化
112
加载不具有属性的MBean
114
100

图1A

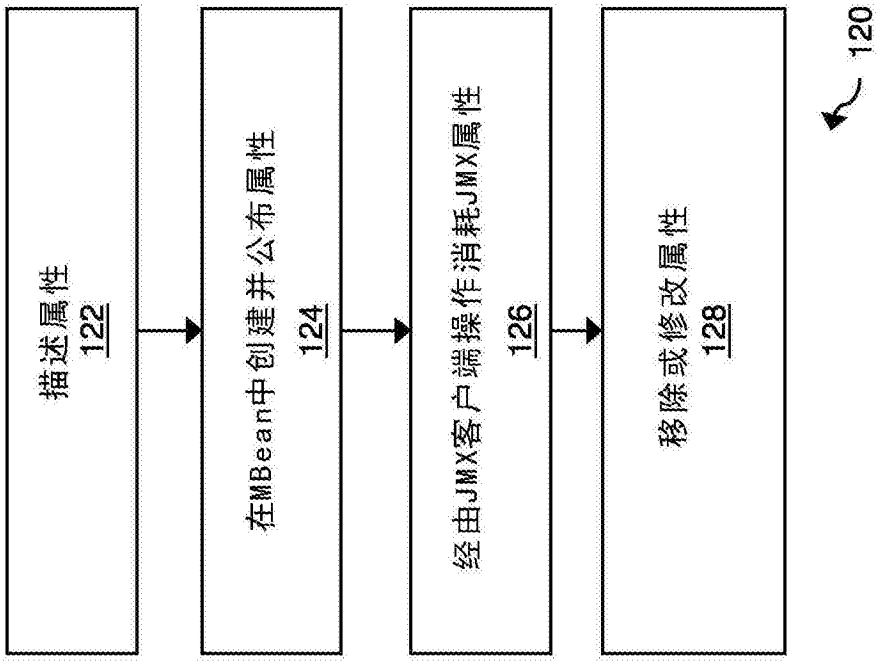

描述属性
122
在MBean中创建并公布属性
124
经由JMX客户端操作消耗JMX属性
126
移除或修改属性
128
120

图1B

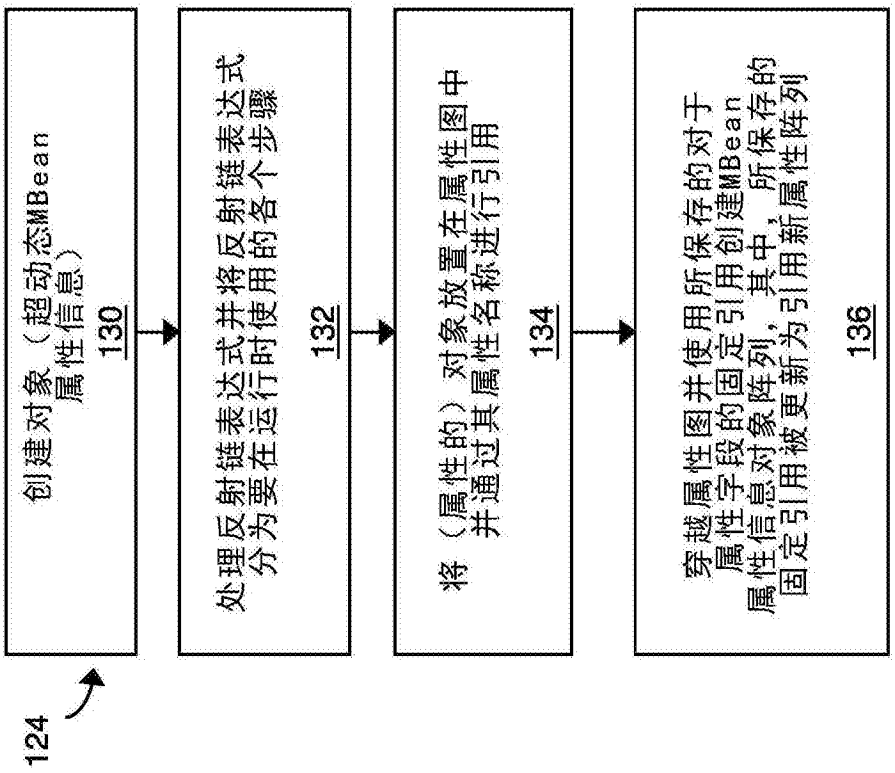

创建对象（超动态MBean
属性信息）
130
处理反射链表达式并将反射链表达式
分为要在运行时使用的各个步骤
132
将（属性的）对象放置在属性图中
并通过其属性名称进行引用
134
穿越属性图并使用所保存的对于
属性字段的固定引用创建MBean
属性信息对象阵列，其中，所保存的
固定引用被更新为引用新属性阵列
136
124

图1C

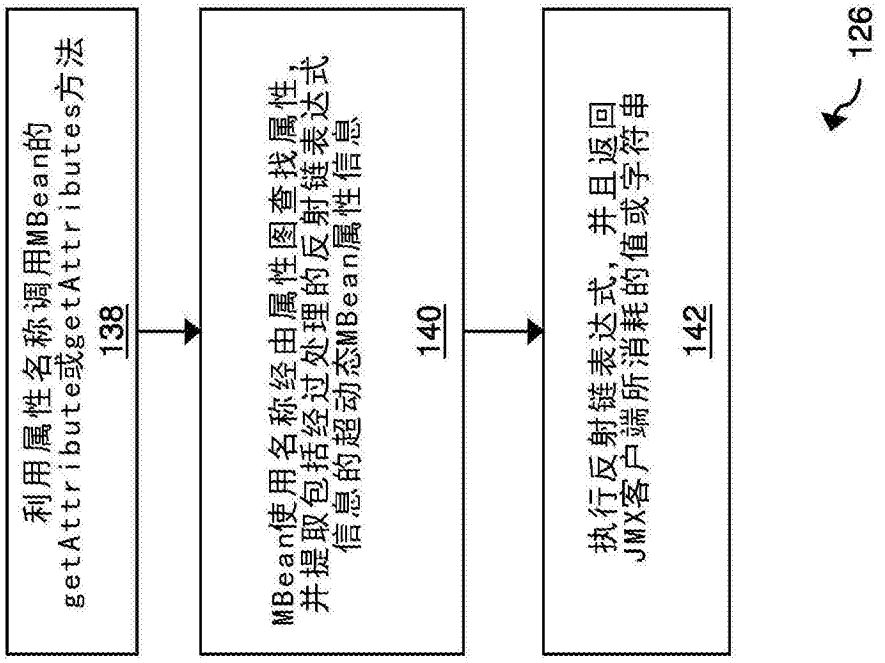

利用属性名称调用MBean的
getAttribute或getAttributes方法
138
MBean使用名称经由属性图查找属性，
并提取包括经过处理的反射链表达式
信息的超动态MBean属性信息
140
执行反射链表达式，并且返回
JMX客户端所消耗的值或字符串
142
126

图1D

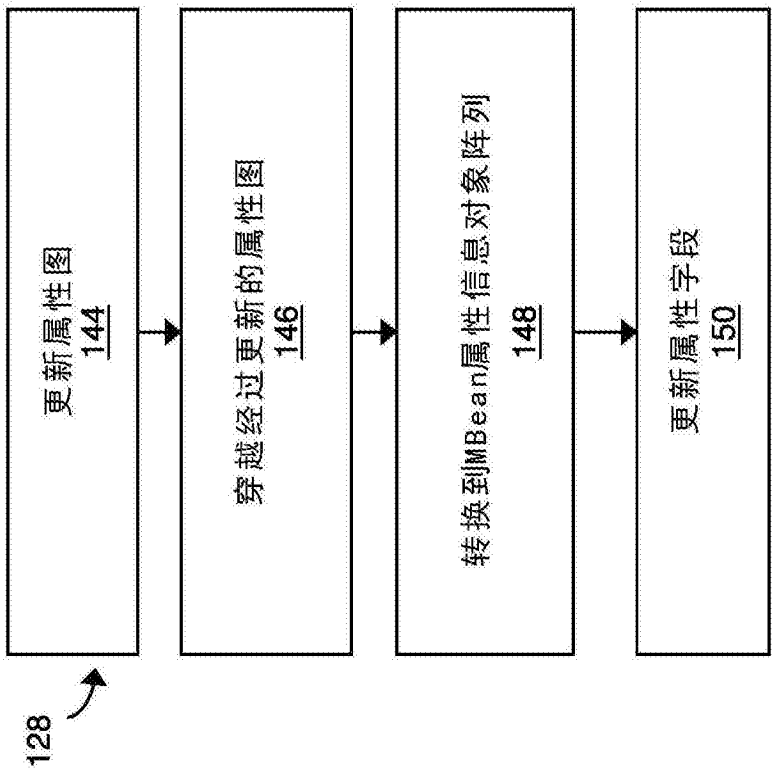

更新属性图
144
穿越经过更新的属性图
146
转换到MBean属性信息对象阵列
148
更新属性字段
150
128

图1E

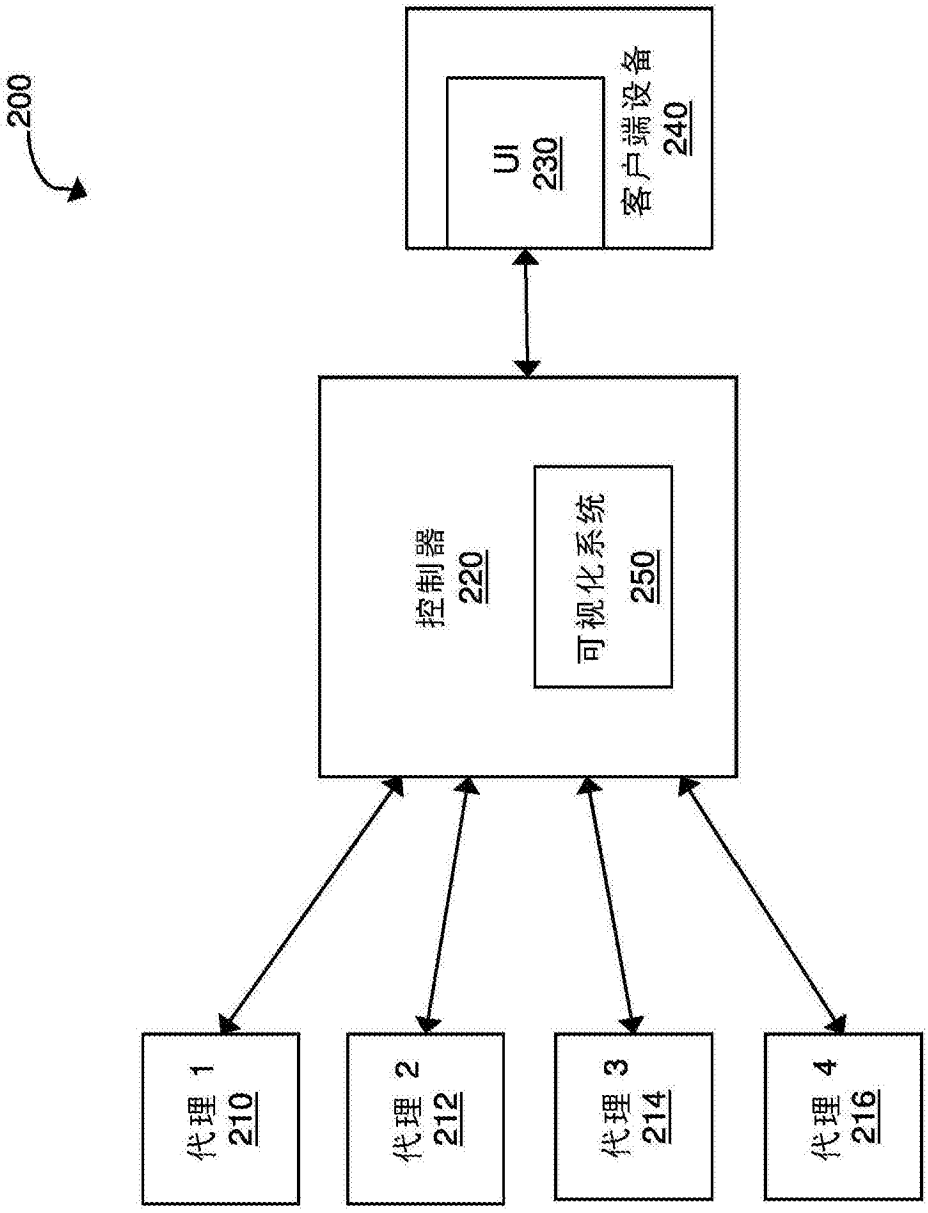

200
客户端设备
240
UI
230
控制器
220
可视化系统
250
代理 1
210
代理 2
212
代理 3
214
代理 4
216

图2

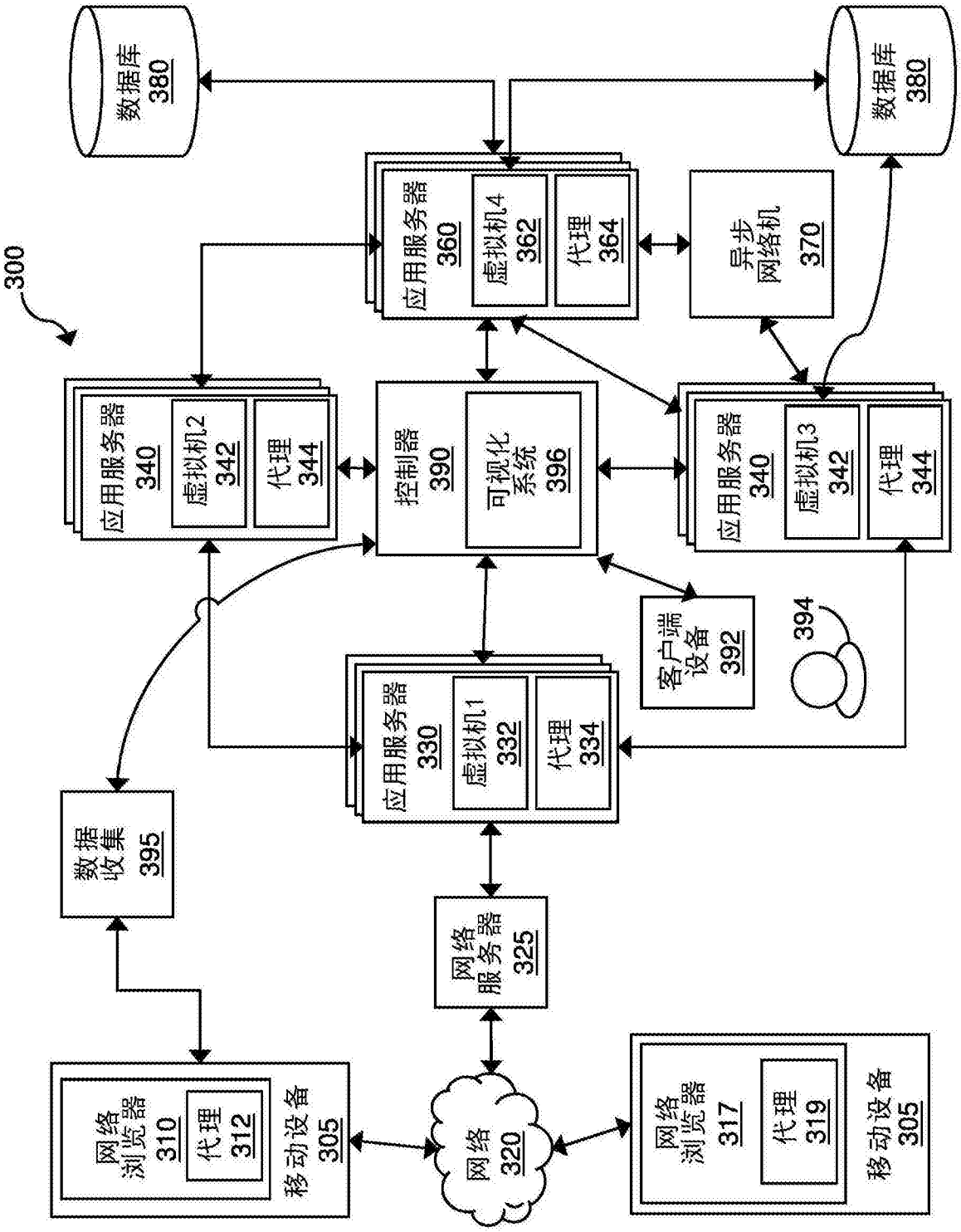

300
数据库 380
数据库 380
应用服务器 360
虚拟机4 362
代理 364
异步网络机 370
应用服务器 340
虚拟机2 342
代理 344
控制器 390
可视化系统 396
应用服务器 340
虚拟机3 342
代理 344
客户端设备 392
394
应用服务器 330
虚拟机1 332
代理 334
数据收集 395
网络服务器 325
网络浏览器 310
代理 312
移动设备 305
网络 320
网络浏览器 317
代理 319
移动设备 305

图3

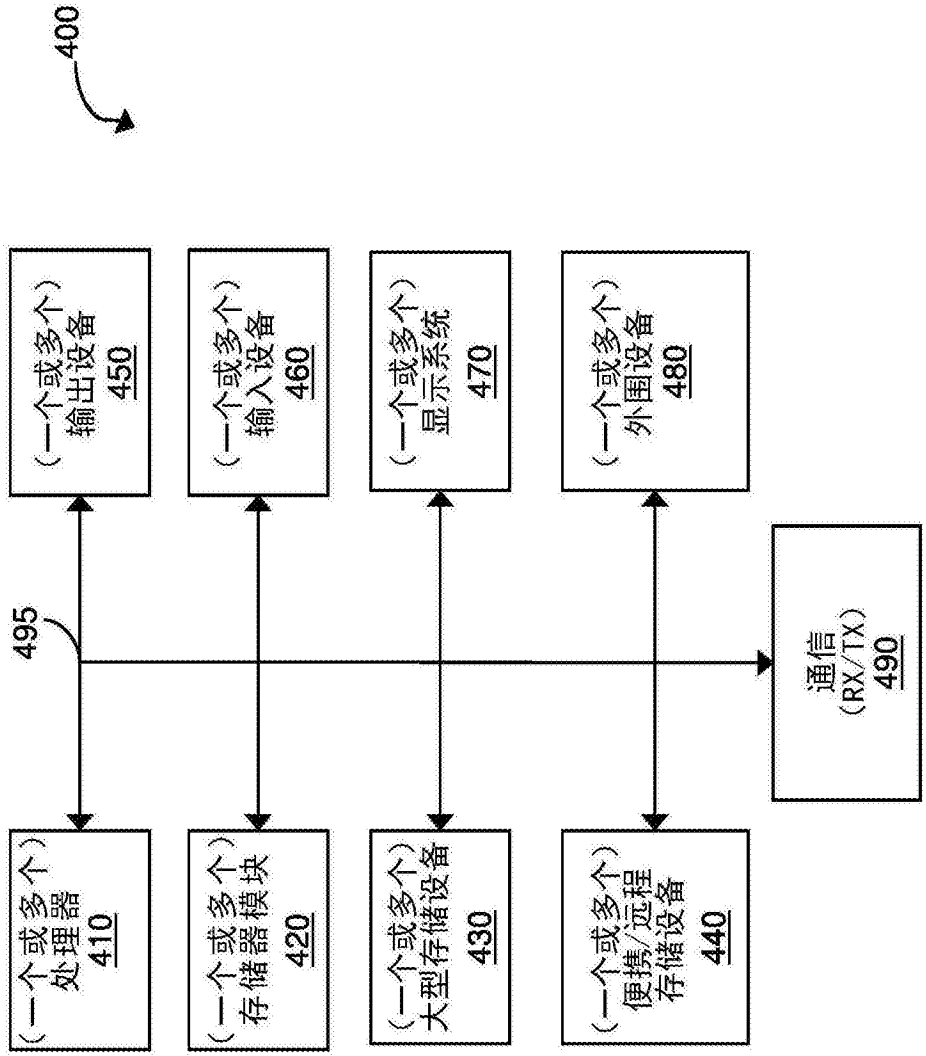

400
（一个或多个）
处理器
410
（一个或多个）
存储器模块
420
（一个或多个）
大型存储设备
430
（一个或多个）
便携/远程
存储设备
440
495
（一个或多个）
输出设备
450
（一个或多个）
输入设备
460
（一个或多个）
显示系统
470
（一个或多个）
外围设备
480
通信
（RX/TX）
490

图4