

(12) 发明专利申请

(10) 申请公布号 CN 102576354 A

(43) 申请公布日 2012. 07. 11

(21) 申请号 201080044371. X

(51) Int. Cl.

(22) 申请日 2010. 07. 30

G06F 15/173 (2006. 01)

(30) 优先权数据

61/230, 584 2009. 07. 31 US

12/718, 924 2010. 03. 05 US

(85) PCT申请进入国家阶段日

2012. 03. 31

(86) PCT申请的申请数据

PCT/US2010/044015 2010. 07. 30

(87) PCT申请的公布数据

W02011/014835 EN 2011. 02. 03

(71) 申请人 电子湾有限公司

地址 美国加利福尼亚州

(72) 发明人 德文达·拉伊库马尔·贾辛加尼

(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 宋鹤

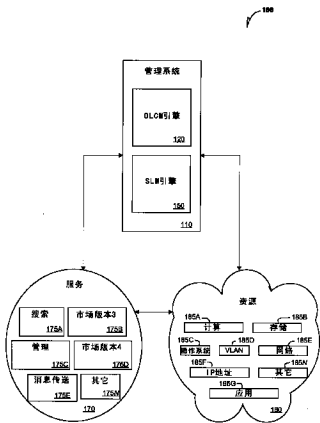
权利要求书 3 页 说明书 9 页 附图 8 页

(54) 发明名称

支持不同部署架构的可扩展框架

(57) 摘要

描述了一种用于管理在多个部署架构间的服务和资源的方法。该方法由定义对应的部署架构的配置文件开始。依据元模型定义每一配置文件。元模型包括：表示通过网络可访问的服务的服务对象，表示由服务消耗的资源资源对象，在服务与资源之间的相互关系，在资源之间的相互关系。针对每一配置文件，基于配置文件生成拓扑。该拓扑包括用于执行任务的资源。



1. 一种方法,包括:

定义对应于部署架构的配置文件,每个配置文件是依据元模型定义的,所述元模型包括:

服务对象,该服务对象表示通过网络可访问的服务,以及  
资源对象,该资源对象表示由所述服务消耗的资源,以及  
在所述服务和资源之间的相互关系,以及

在所述资源之间的相互关系;以及

针对每一配置文件,

使用一个或者多个处理器,基于所述配置文件生成拓扑,所述拓扑包括用于执行任务的资源。

2. 依据权利要求 1 所述的方法,其中,所述元模型还包括配置对象,所述配置对象中的每一个与所述资源对象的一个资源对象相关联,所述配置对象声明与所述一个资源对象相关联的配置。

3. 依据权利要求 1 所述的方法,其中,所述元模型还包括性能对象,所述性能对象中的每一个与所述资源对象相关联,所述性能对象声明所述资源对象的性能度量。

4. 依据权利要求 1 所述的方法,其中,所述元模型还包括能力对象,能力对象中的每一个与所述资源对象相关联,所述能力对象声明所述资源对象的能力度量。

5. 依据权利要求 1 所述的方法,其中,依据用于该架构域和另一个架构域的基模型定义所述配置文件的至少一部分。

6. 依据权利要求 1 所述的方法,其中,所述服务对象与另一个配置对象,另一个性能对象,以及另一个能力对象相关联。

7. 依据权利要求 1 所述的方法,其中,所述资源对象是硬件资源或者软件资源,或者互联网协议 (IP) 地址,或者虚拟局域网 (VLAN),以及所述硬件和软件资源对象包括多个部件对象。

8. 依据权利要求 1 所述的方法,其中,所述元模型包括群组对象,所述群组对象标识所述资源对象中的一个或者多个,或者所述部件对象中的一个或者多个,所述群组对象具有相关联的性能对象和相关联的配置对象。

9. 依据权利要求 1 所述的方法,还包括对在所述拓扑中指定的服务的管理,所述管理是通过以下方式进行的:

管理在所述拓扑内的所述资源中的一个资源,该一个资源对应于所述资源对象,以及

基于在所述拓扑内的该一个资源的配置文件,管理在所述拓扑内对应于该一个资源的物理组件。

10. 一种系统,包括:

管理系统,该管理系统由一个或者多个处理器实现以访问在部署架构内的服务的一个或者多个配置文件,并且基于配置文件生成拓扑,该配置文件是依据元模型定义的,所述元模型包括:

服务对象,该服务对象表示所述服务中的一个服务,以及

资源对象,该资源对象表示由所述服务的至少一部分消耗的资源,以及  
在所述一个服务和资源之间的相互关系,以及

在所述资源之间的相互关系;以及

服务管理器,该服务管理器用于管理所述一个服务的操作生命周期和服务水平,所述一个服务对应于所述部署架构,所述服务管理器被配置为动态地加载如下两者:

控制器,该控制器管理在所述拓扑内的所述资源中的一个资源,所述一个资源对应于所述资源对象,以及

适配器,该适配器将命令从所述控制器翻译为被管理的资源的实例。

11. 依据权利要求 10 所述的系统,其中,所述元模型还包括配置对象,所述配置对象中的每一个与所述资源对象中的一个资源对象相关联,所述配置对象声明与所述一个资源对象相关联的配置。

12. 依据权利要求 10 所述的系统,其中,所述元模型还包括性能对象,所述性能对象中的每一个与所述资源对象相关联,所述性能对象声明所述资源对象的性能度量。

13. 依据权利要求 10 所述的系统,其中,所述元模型还包括能力对象,所述能力对象中的每一个与所述资源对象相关联,所述能力对象声明所述资源对象的能力度量。

14. 依据权利要求 10 所述的系统,还包括配置管理器,用于将所述拓扑和一个或者多个配置文件存储在存储器存储设备内,并且核实在所述拓扑内的资源的配置。

15. 依据权利要求 10 所述的系统,还包括资源管理器,用于分配由所述服务中的一个服务消耗的资源。

16. 依据权利要求 10 所述的系统,还包括微内核,用于生成和控制所述服务管理器。

17. 依据权利要求 10 所述的系统,其中,依据基模型定义所述配置文件,所述基模型包括部署架构间共同的元件和仅专用于给定的部署架构的元件的另一集合。

18. 依据权利要求 17 所述的系统,其中,所述元模型包括群组对象,所述群组对象标识所述资源对象中的一个或者多个,所述群组对象具有相关联的性能对象和相关联的配置对象。

19. 依据权利要求 10 所述的系统,其中,资源实例是在数据中心中。

20. 依据权利要求 10 所述的系统,其中,物理组件是在云计算环境中。

21. 依据权利要求 10 所述的系统,其中,所述资源对象包括硬件资源,软件资源,互联网协议 (IP) 地址,以及虚拟局域网 (VLAN) 中一个或者多个。

22. 依据权利要求 21 所述的系统,其中,所述硬件资源和所述软件资源还包括部件。

23. 依据权利要求 22 所述的系统,其中,对应于所述硬件资源的所述资源包括计算元件,网络元件,以及存储元件。

24. 依据权利要求 22 所述的系统,其中,对应于所述软件资源的所述资源包括操作系统和应用。

25. 一种计算机可读存储介质,其上包含有指令,所述指令是一个或者多个处理器可执行的,用于执行一种用于在多个部署架构之间管理服务的方法,所述方法包括:

为每一部署架构定义配置文件,所述配置文件是依据元模型定义的,所述元模型包括:

服务对象,该服务对象表示一个或者多个服务中的一个服务,以

及

资源对象,该资源对象表示由所述一个服务消耗的资源,以及

在所述一个服务和资源之间的相互关系,以及

在所述资源之间的相互关系;以及

针对每一配置文件,使用一个或者多个处理器,基于所述配置文件生成拓扑,所述拓扑包括用于执行任务的资源。

## 支持不同部署架构的可扩展框架

[0001] 交叉参考

[0002] 本申请要求申请于 2009 年 7 月 31 日, 题目为“Management of Services and Resources”的美国临时申请第 61/230, 584 号的优先权, 以及申请于 2010 年 3 月 5 日, 题目为“Extensible Framework to Support Different Deployment Architectures”的美国申请第 12/718, 924 号, 它们的全部通过引用被结合于此。

### 技术领域

[0003] 本申请一般地涉及在数据中心或者云计算环境中的服务和资源的管理, 以及在一个具体的示例中, 涉及支持不同部署架构的可扩展框架。

### 背景技术

[0004] 在数据中心环境中, 存在完成商业目标或者支持其它服务的许多服务。在一些上下文中, 这些服务可以包括交易 (卖物品, 查看出售的物品等), 消息传送和搜索。这些服务都消耗硬件和软件资源。硬件资源的示例包括计算机 (服务器), 网络 (交换机, 负载均衡器, 防火墙等), 存储 (SAN, NAS 等), 软件资源的示例的包括操作系统, 应用平台堆 (java 虚拟机, tomcat servlet 容器等), 以及数据库。将这些硬件和软件资源根据具体服务的需求安排在不同的配置中。配置被称为“部署架构”。部署架构的示例包括传统三层架构 (网络层, 应用层, 数据库层, 它们的每一个可以具有用于流量分布的负载均衡器), 消息传送基础结构等。在这些内可以存在变化, 例如, 可以将负载均衡器成对地部署以提供高可用性 (HA)。传统上, 存在针对数据中心环境的每一子集的管理系统。例如, 存在关注管理 (例如, 配置, 健康检查, 性能监视等) 交换机的网络管理系统。其它关注应用 (例如, 应用部署等)。这导致管理系统的激增。

### 附图说明

[0005] 在附图中通过举例而非限制图示了一些实施例, 在附图中:

[0006] 图 1 是在其中可实践多种实施例的网络环境的框图。

[0007] 图 2 是依据多种实施例的管理系统的框图。

[0008] 图 3 是依据多种实施例用于使用在域布置中的模型框架的处理的流程图。

[0009] 图 4 是依据多种实施例在资源分类内的对象的层次结构。

[0010] 图 5 是依据一些实施例的硬件和软件资源的模型结构的框图。

[0011] 图 6 是依据多种实施例与群组相关联的模型架构的框图。

[0012] 图 7 是依据多种实施例用于存储数据的模型结构的框图。

[0013] 图 8 是在其内可以执行指令集的机器的图表表示, 该指令集用于令机器执行在这里讨论的任何一个或者多个方法。

[0014] 图 9 是依据多种实施例的配置文件 (profile) 创建器的框图。

[0015] 图 10 是依据一些实施例构建在两个层中的信息模型的框图。

## 具体实施方式

[0016] 描述了使用可扩展模型框架管理在数据中心环境中的服务和资源消耗资源的方法和系统。在以下的描述中,针对说明的目的,为了提供示例实施例的彻底地理解,陈述了许多具体的细节。然而,对于本领域的技术人员将显而易见的是可以没有这些具体的细节地来实践本发明。对在这里使用的,可以将以术语“或者”理解为包括或者排除的意义。

[0017] 在大型企业计算系统中,使用不同系统来提供服务。该系统的示例提供交易服务,搜索服务,消息传送服务,支付服务以及网络发布服务。因为每一系统执行单独的功能,所以作为整体的企业计算系统的操作依赖其它系统的每一个的性能。在使用多于一个系统来执行贸易或者“流程”时这尤其正确。例如,为了完成在英特网商务系统中的销售,在流程中的特定时刻可以使用对应于搜索,网络发布,支付和消息传送服务的系统。

[0018] 使用一个或者多个资源实现系统。系统资源可以是硬件资源和/或软件资源。硬件资源的示例包括服务器,路由器,负载均衡器等。软件资源的示例包括操作系统,应用等。

[0019] 在管理系统的上下文中,将每一被管理的实体(即,服务或者资源)以它的操作配置,性能和状态(例如,活动状态)的形式来建模。此外,每一该建模的实体与一个或者多个建模的实体有关以反映现实世界关系。例如,计算机程序(应用)在操作系统上运行,并且因此在管理系统内的对应的建模的实体也反映这个关系。所产生的建模的实体(每一实体由指定它的配置,性能和状态的属性组成)的模型和它们的相互关系的模型称为信息模型。在建模工具中创建该信息模型并将信息模型存储在模型库中以例如由系统设计师来使用。

[0020] 每一服务类型具有部署架构,该部署架构示出多种构成资源,它们的相互关系以及它们与其它服务的相关性(如果适用的话)。在模型驱动的管理系统的上下文中,可以以具体的信息模型的形式指定每一服务的部署架构。在该上下文中,称信息模型为部署配置文件(“配置文件”)。因此使用该配置文件以其功能的和非功能的准则的形式来指定部署架构。例如,最简的配置文件可以是“N个服务器”,其中可以将服务器连续地指派。另一配置文件可以对应于传统的三层架构,该三层架构包括:负载均衡器,要用作网络服务器的服务器,要用作应用服务器的服务器,要用作数据库服务器的服务器,以及提供 LAN 交换和在 VLANs 之间的第三层路由的第三层交换机。这三种类型的服务器可以具有不同的能力(例如,数据库服务器可以为大铁箱,网络服务器可以为刀片服务器或者甚至是虚拟机)。其它变化可以导致不同的配置文件,例如,可以将负载均衡器安排为主动—主动(或者主动—被动)配置以提供高可用性。

[0021] 另一配置文件可以使用三层配置文件以生成另一服务的配置文件。例如,配置文件可以指定网络服务器(软件)将要被部署在层 1 服务器上,应用服务器(软件)将要被部署在层 2 服务器上,以及数据库将要被部署在层 3 服务器上等。在这些配置文件中,在应用服务器中要部署的应用代码可以依赖其它服务(例如,日志记录服务)。

[0022] 在数据中心或者云计算环境中的多种部署架构具有诸如服务器,负载均衡器,操作系统等之类的许多共同的资源类型。为了让单个管理系统理解多个部署架构,在配置文件间重用共同的资源类型定义。这通过如图 10 示出的那样在两个层中构建信息模型来实现。基模块 1002 提供诸如服务器,操作系统等之类的共同的资源类型的定义,而顶层提供

专用于诸如交易 1004, 消息传送 1006, 搜索 1008, 支付 1010, 网络发布 1012 之类的特定服务的资源和服务类型的定义。接着通过将特定域的服务和资源类型与在基模块 1002 中的类型相结合来构建配置文件。

[0023] 当要部署服务时, 关于如下方面来指定它的部署配置: 资源的数目和类型, 单独的和 / 或根据类型的资源的配置参数, 接着将合适的资源实例进行识别和分配。这样做允许管理系统将部署和其它生命周期管理任务自动化。在本发明的上下文中, 在被称为部署拓扑 (以下称为拓扑) 的规约中捕获该信息。拓扑是配置文件的实现。拓扑的规约在两个级中进行: 逻辑的和物理的。逻辑拓扑指定资源和它们的基数 (例如, 10 个刀片服务器) 的抽象类型, 并且物理拓扑指定资源的具体类型和它们的具体实例 (例如, 有资产识别码 54321 abcde 的 IBM LS 20 刀片服务器)。

[0024] 管理系统管理对应于不同配置文件的服务, 允许新配置文件的引入, 并且因此随着数据中心或者云计算环境的进化而管理新的服务。此外, 其不修改管理系统的子系统, 或者重部署管理系统或者重启管理系统的运行地来实现此。可以通过提供通用服务管理器, 元模型和插件框架来实现这些目标。

[0025] 服务管理器负责服务的完整生命周期和服务水平管理。生命周期操作的示例包括部署, 配置, 启动, 激活, 更新等, 并且可以作为一个整体在服务上实施这些操作, 或者在由服务使用的具体的或者成组的资源上实施这些操作。服务管理器的每一实例可以管理遵从同一配置文件的多个服务实例。具体性在于专用于抽象资源类型 (例如, 负载均衡器) 的控制器中, 以及在于专用于具体资源类型 (例如, Citrix NetScaler 负载均衡器) 的适配器中。当首先部署服务时, 服务管理器分析拓扑并且针对指定的每一独特的服务和资源来动态加载相对应的控制器和相关联的适配器。

[0026] 在服务管理器内的每一控制器管理同一服务或者资源类型的多个实例, 这是因为它是针对每一抽象类型而定义的。例如, 使用单个负载均衡器控制器来管理服务中的全部负载均衡器而不管使用的负载均衡器的特定样式或型号。适配器专用于具体类型 (例如, 特定的样式或型号) 并且将通用命令翻译为专用于实现方式的命令。例如, 存在 NetScaler 负载均衡器适配器, F5BigIP 负载均衡器适配器等。像这样, 为了引入对 Zeus 负载均衡器的支持, 仅提供它的适配器。为了引入例如 SSL 加速器的新类型, 提供针对具体类型的控制器和适配器 (例如, 用于 Coyote Point's SSL 加速器的适配器)。以上引用的示例是硬件专用的, 但是该概念也适用于软件类型。

[0027] 对应于多种资源和服务类型的信息模型元件和相关联的管理语义被局限于在服务管理器内部的控制器和适配器。管理系统的剩余部分 (包括除了控制器和配置器之外的服务管理器组件) 在信息模型的较高的抽象水平上操作。称该抽象为元模型。因为子系统将要在元模型水平上操作, 所以只要配置文件遵从元模型, 它们就不受在现有配置文件中做出的改变或者新配置文件的引入的影响。因此, 使用元模型允许新服务的管理, 该新服务的配置文件可能包括新模型元件 (例如, 诸如 SSL 加速器之类的新类型)。

[0028] 元模型定义了以下八个元对象: 服务, 资源, 部件, 配置, 性能, 能力, 群组和应用数据。资源元对象还具有子类型。元模型还指定这些元对象之间的关系。将在配置文件中的每一类型分类 (即, 子类型化) 为元对象之一。例如, 将在配置文件中的每一资源子类型化为元模型中的资源子类型之一。作为一个具体的示例, 将负载均衡器, 交换机, 路由器, 防火

墙从网络元件子类型化,进而,该网络元件是从硬件资源子类型化的。进而,将该硬件资源从资源子类型化。为了引入例如 SSL 加速器的新实体,可从网络元件(通过概括也使它成为硬件资源和资源)导出该新实体。

[0029] 可以将元模型视为用于创建配置文件的模型。希望引入针对新服务的管理能力的用户或者域专家(例如,应用设计师)可以检查模型库并且从可用的资源类型中进行选择,以及创建(如果需要)专用于他的域的新资源或者服务类型。将新创建的类型从现有的元对象或者子类型中子分类出来。接着,域专家可以创建在多种服务和资源类型之间的适当的关系和基数,从而创建新配置文件。接着将该配置文件编排版本并存储在管理系统中。

[0030] 为了让管理系统使用拓扑中所指定的配置,基数,资源绑定和关系信息,该管理系统内部创建类型的程序化表示。这是通过从配置文件创建类定义实现的。当指定拓扑为输入时,管理系统可以生成对应的对象,可以从该对象获得所述信息。在一种示例实现方式中,可以通过 XML 模式和 XML 实例文档的拓扑表示配置文件。接着使用 JAXB 框架,可以生成 Java 类表示,并且在运行时,可以分析拓扑以创建对应的 Java 对象。

[0031] 提供了使管理系统能够动态加载能力以管理可能具有新部署架构的新服务的插件框架。具体地,该插件框架允许将配置文件中的模型元件的类表示,和所需要的对应的控制器和适配器捆绑到库中。将如此创建的库称为插件,并且使用插件框架,将该库存在在管理系统中。将类和插件编排版本以防止动态类/库加载错误。插件框架维护在[配置文件名称,配置文件版本]二元组和[插件名称,插件版本]二元组之间的映射。当必须部署服务时,服务管理器检查其中存储了对应的配置文件的名称和版本号的数据。使用该信息,服务管理器识别有正确版本的对应的插件,并且使用可用语言和操作系统设施将它动态地加载到它的地址空间中。这是在不需重新部署管理系统或者不用重启正在运行的管理系统的情况下完成的。另外,因为系统的剩余部分操作在元模型的水平上,所以不需要做出子系统修改。因此,使用通用服务管理器,元模型和插件框架,实现了可扩展性。

[0032] 作为一个示例,不是开发他们自己的管理系统以管理搜索基础结构(由多个服务和资源构成),搜索领域专家(或者“搜索团队”)可以利用可扩展模型的框架能力(监视,配置管理,资源管理等),并且仅添加专用于搜索域的那些部分。更具体地,搜索团队可以重用服务器,负载均衡器,操作系统等的定义来建立它们的配置文件,并且引入例如搜索查询节点(软件资源)的新元件。搜索团队可以使它们的(一个或者多个)配置文件与元模型一致。针对新类型,搜索团队可以写相关的控制器和适配器。搜索团队可以创建将模型元件的类定义封装在配置文件中的适当插件,以及所需要的相关的控制器和适配器。接着可以对该插件编排版本并存储在管理系统中,并且当必须部署对应的服务时将该插件动态地加载。

[0033] 图 1 是诸如数据中心之类的示例性基础结构环境的框图 100,并且示出其包括服务 170,资源 180 以及管理系统 110 之间的关系。服务 170 例如可以包括搜索服务 175A,版本 3 市场服务 175B,管理服务 175C,版本 4 市场服务 175D,消息传送服务 175E,以及使用资源且可被管理的任何其它服务 175N。资源 180 例如可以包括计算机资源 185A,存储资源 185B,操作系统资源 185C, VLAN 资源 185D,网络资源 185E, IP 地址资源 185F,应用资源 185G,以及服务 170 可以使用的任何其它资源 185N。在示例实施例中,管理系统 110 包括操作生命周期 (OLCM) 引擎 120,用于部署服务 170 使得它们使用从资源 180 选择的资源,

等等。在示例实施例中,管理系统 110 还包括服务水平管理 (SLM) 引擎 150,用于监视服务 170 和资源 180,并且动态地分配资源 180 中的至少一者使得每一服务 170 维持关键性能指示 (KPI) 所定义的指定的服务水平,等等,该服务水平诸如平均响应时间或者吞吐量。

[0034] 在示例实施例中,服务 170 和资源 180 耦接到管理系统 110,使得管理系统 110 使用 OLCM 引擎 120 和 SLM 引擎 150 可以管理服务 170 和资源 180。在示例实施例中,可以将服务 170 的一些与资源 180 的一个或者多个耦接,使得为服务 170 的一些将资源 180 的一个或者多个进行分配,或者部署服务 170 的一些使得它们使用资源 180 的一个或者多个。

[0035] 图 2 是依据多个实施例的管理系统 102 的框图。管理系统 102 包括可选的配置文件创建器 202,一个或者多个服务管理器 204,微内核 210,分配器 212,配置管理器 214,供应管理器 216,资源管理器 218,以及锁定管理器 220。管理系统 102 提供针对被管理的服务和构成资源的操作生命周期管理,动态资源分配,以及服务水平管理能力。

[0036] 现在参考图 9,描述了示例配置文件创建器 202。在某些实例中,配置文件创建器 202 可以在管理系统 102 的外部。配置文件创建器接收来自用户 902 (例如,域专家) 的一个或者多个输入并且访问模型库 904。模型库 904 存储服务 and 资源的元模型和模型。用户 902 适当地重用来自模型库 904 的模型元件以部署被创建的服务类型的架构。如果需要创建新模型元件,则用户 902 在该工具中创建它们的定义。此外,将每一新创建的模型元件从元对象或者从资源子类型之一适当地子分类。可以按照需要创建模型元件之件的相关性和亲子关系。如果在目标配置文件和其它配置文件之间存在相关性 (即,目标服务类型依赖另一服务类型),那么也可以创建该相关性。在一个示例实施例中,可以以 UML 形式创建部署架构的模型,并且接着将 UML 表示变换为 XML 模式表示。接着,使用 JAXB 可以将 XML 模式编译以产生 Java 类表示。一旦创建了配置文件,接着将其从工具导出并且编排版本以及存储在管理系统中。

[0037] 当创建并部署了新服务实例时,用户 (例如,操作设计师) 使用拓扑编辑器来创建拓扑。如此创建的拓扑是逻辑拓扑。逻辑拓扑指示抽象的资源类型 (例如,“服务器”,“负载均衡器”,“交换机”等) 以及将使用资源的每一个的多少 (例如,基数)。接着,在两步处理中为抽象资源创建资源绑定——将抽象资源类型绑定到具体资源类型 (例如,“NetScaler 负载均衡器”,“Cisco 3550 交换机”,“LS20 服务器”等),接着绑定到实际的资源实例。这导致物理拓扑的创建。在某些实例中,当部署了服务时,管理系统 102 可以缓慢地将具体的资源类型绑定到实际资源实例。

[0038] 服务管理器 204 依据数据中心 104 或者云计算环境 106 内的配置文件来管理服务 and 构成资源。更具体地,服务管理器 204 为服务和构成资源提供服务水平管理 (SLM) 和操作生命周期管理 (OLCM)。基于物理拓扑,服务管理器 204 启动,管理,和 / 或终止在数据中心 104 和 / 或云计算环境 106 中的实际资源的执行以提供服务。

[0039] 服务管理器 204 包括专用于服务或者资源类型的控制器 206。它负责在它的控制下全部服务 / 资源实例的完整生命周期和服务水平管理。这样,每一服务管理器 204 可以包括针对消耗多于一种资源类型的服务的多于一个控制器 206。

[0040] 服务管理器 204 还包括适配器 208,该适配器 208 提供如下转换:从控制器 206 接收的命令到 (一个或者多个) 具体资源实例可执行的本地命令。控制器 206 针对每一具体资源类型可以访问不同的适配器 208。可以使用单个适配器以与同一具体资源类型的多于

一个的实例进行通信。

[0041] 微内核 210 为每一子系统（例如，服务管理器 204，分配器 212，配置管理器 214，供应管理器 216，资源管理器 218，以及锁定管理器 220）提供生命周期和服务水平管理。它还提供服务注册能力和服务查找能力以分别注册和查找子系统的服务端点。

[0042] 分配器 212 作为到管理系统 102 的进入点来服务。它接收全部的客户请求，通过在微内核 210 中进行查找来判定哪个子系统作为请求的目标，并且接着将请求分配到目标子系统。它还为用户提供用户认证和授权，并且创建和维护用户会话。可以将用户认证降级到不同于管理系统 102 的另一服务器。可以基于特定用户的角色和特权内部地执行用户认证。在成功地认证和授权后，创建用户会话，并且该用户会话持续直到经过一段时间周期或者当用户退出时。

[0043] 配置管理器 214 可以存储和 / 或访问现有的配置文件和 / 或存储在配置库的它们的对应的拓扑（例如，逻辑的或者物理的）。在一些实例中，可以使用关系数据库管理系统（RDBMS）实现该配置库，或者该配置库可以是配置管理数据库（CMDB）。

[0044] 配置管理器 214 可以操作在元模型水平而不是在配置文件中的模型元件水平，以保持与单独的部署架构 / 模型的独立性。配置管理器 214 基于对应的元对象可以将拓扑（例如存储为 XML 文档）转换为关系数据库表。配置管理器 214 还可以为在拓扑中的配置文件，拓扑（逻辑的和物理的）以及单独的对象创建和维护版本向量，以允许管理系统 102 将服务回滚（或者向前）到它的部署的拓扑的之前（或之后）的版本。配置管理器 214 还可以核实部署的配置并且保证在资源中不存在配置漂移。当检测到配置漂移时，受影响的资源的相关的服务管理器 204 可以采取适当的正确的行动。

[0045] 供应管理器 216 负责资源的部署，资源包括用于计算元件（例如，服务器）的应用代码，应用堆（例如，Java 虚拟机（JVM），Servlet 容器）和操作系统。因为多种专门化产品和工具是可用的，所以供应管理器 216 提供服务层以使用适配器 208 抽象那些产品和工具。该工具的示例包括但不限于用于 OS 部署的 Symantec 的 Altiris 和用于应用堆和代码部署的 eBay 的 TurboRoller。

[0046] 资源管理器 218 负责保留和分配实际的，具体的资源。保留是被租用的，即，如果在特定的时间内特定的服务没有使用，那么保留到期。可以通过基于用户（例如，系统管理员）发出的服务部署命令执行资源分配使保留是永久的（或者到达指定的最大持续时间）。资源管理器 218 还可以动态地指派资源并且周期地调节服务间的资源分配。

[0047] 锁定管理器 220 可以操作以防止同时访问共享的资源。在一些实例中，在不同服务之间资源被共享。如果多个服务管理器 204 同时访问共享的资源，那么有可能创建在资源配置中的不一致性，这是因为不是全部的资源施行或者提供串行化访问。因此，可以提供带外同步或者互斥机制。为了访问资源，服务管理器 204 首先从锁定管理器 220 获取锁定，并且在它与管理的资源的会话结束之后释放锁定。锁定是被租用的，即，在指定的持续时间之后它们到期。依据需要，可在某可调的最大值之内延长该租用。锁定还是在重启前后可重新进入并且不变的。

[0048] 图 3 是用于创建配置文件和拓扑的处理 300 的流程图。每一服务（例如，消息传送或者交易等）具有它的相对应的部署架构。在步骤 302 处以及如以上所描述的，将该部署架构识别或者创建。

[0049] 然而,该架构是软件程序不能使用的。因此,创建软件可以使用的配置文件。在步骤 304 中,使用元模型和现有的模型库(该库将包括基模块和可能的可以重用的其它用户贡献的模型元件)来创建该配置文件。如果需要也可以创建新模型元件(通常是域专用的)。接着将该配置文件编排版本并且存储在管理系统中。

[0050] 在步骤 306 中,当要部署新服务实例时(例如,消息传送系统的新实例),基于用户提供的参数(例如,服务器数目,配置参数等)或者通过某种策略来创建它的(部署)拓扑。如此创建的拓扑是逻辑拓扑。下一步是进行资源绑定,资源绑定的结果是创建了物理拓扑。将逻辑的和物理的拓扑两者都编排版本并存储在管理系统中(更精确地,在配置管理器中)。可使用拓扑编辑器可视地或者通过另一程序程序化地创建该拓扑。一旦接收到全部的预期的准许,则部署物理拓扑(更精确地,管理系统使用物理拓扑以部署服务实例)。

[0051] 图 4 是用于依据多种实施例对在元模型内的被用于分类资源的对象 400 的层次结构。该层次结构一般地由不同类型的资源组成。

[0052] 资源 402 可以被分类或者子类型化为一个或者多个资源类型。在该模型的本示例中的这些资源被描述为在资源 402 下方的第二层,并且包括硬件资源 404, IP 地址 406, 虚拟局域网(VLAN) 406 和软件资源 410。元模型可以允许更多的资源分类。

[0053] 还可以将在元模型中的硬件资源 404 和软件资源 410 分类为子类型。在图 4 的第三和第四层中描述了这些子类型。图 4 中的这些子类型不意图成为在元模型中的限制。硬件资源 404 可以被子类型化为计算元件 412(例如,服务器,虚拟机,计算机,或者手持式设备),或者网络元件 414(例如,路由器,负载均衡器,防火墙等),或者存储元件 416(例如,存储器存储设备,光存储设备,存储局域网,网络附接的存储等)。软件资源 410 可以被子类型化为操作系统 418(例如,Windows, Linux 等),或者应用 420(例如,浏览器或者字处理)。应用 420 还可以被子类型化为应用容器 422(例如,Java 虚拟机,Servlet 容器等)。

[0054] 图 5 是依据一些实施例示出在元模型中的硬件和软件资源的合成的框图。硬件资源 404 和软件资源 410 两者都具有描述它们的配置 502, 性能 504 和能力 506 的对象。在配置对象 502 中的信息被用于配置资源实例。配置对象 502 自身可以由子配置对象构成(未示出)。通过监视相对应的资源实例的性能度量而获取的信息被存储在性能对象 504 中。存储在能力对象 506 中的信息描述了该资源类型的能力(例如, CPU 的数目, 存储器的量, 吞吐量, 响应时间等)。可通过如下方式来在自动配对资源分配期间使用存储在能力对象 506 中的信息:指定资源类型所需要的能力并且将它与实际的具体类型提供的能力进行匹配。硬件资源 404 和软件资源 410 两者都可以分别由硬件部件 508(例如,网络接口卡)和软件部件 510(例如,动态链接库)建立。每一部件还可以具有一个相关联的能力对象(未示出)。一些部件自身可以由子部件(硬件子部件 512 和软件子部件 514)构成,等等。可以将硬件部件连接到另一远程硬件部件 516 用于访问远程硬件资源 518。例如,可以通过外部线缆将在服务器(硬件资源 404)上的网络端口(硬件部件 508)连接到在网络交换机(远程硬件资源 518)上的网络端口(远程硬件部件 516)。

[0055] 图 6 是依据多种实施例与群组对象 602 相关联的模型结构 600 的框图。在一些实施例中,将资源 604(例如,硬件资源 404 和软件资源 412)和/或部件 606 编组以使能群组化操作。例如,可以将服务器(其是计算元件)编组到一个池中,并且可以将连续的 IP 地址编组到一个子网中。可以经由群组对象 602 提供编组。每一群组可选地具有相关联的配置

对象 610 和性能对象 608。

[0056] 图 7 是依据多种实施例在元模型中的应用数据 702 的模型结构 700 的框图。可以将应用数据 702 存储在与操作系统 418 相关联的文件系统中,并且由应用 420 提供服务或者由应用 420 使用。应用数据 702 的示例包括数据库表和用于包括网页的内容的数据。

[0057] 图 8 是在其中可以执行指令集的机器的图表表示,该指令集可以执行用于令机器执行在这里讨论的方法的任何一个或者多个。在可替代的实施例中,机器作为独立设备来操作或者可以将机器连接(例如,网络连接)到其它机器。在网络连接的部署中,机器可以以在服务器-客户机网络环境中的服务器或者客户机的功能来操作,或作为在对等(或者分布的)网络环境中的对等机器。该机器可以为服务器计算机,客户计算机,个人计算机(PC),桌面 PC,机顶盒(STB),个人数字助理(PDA),蜂窝电话,网络应用,网络路由器,交换机或者桥,或者能够执行指定了该机器要采取的行动的指令集(连续的或者其它)的任何的机器。另外,虽然仅图示了单个机器,术语“机器”还应当包括任何单独地或者联合地执行指令集(或者多个集)以执行在这里讨论的任何一个或者多个方法的机器的任何的集合。

[0058] 示例计算机系统 800 包括处理器 802(例如,中央处理单元(CPU),图形处理单元(GPU),或者它们二者),主存储器 804 和静态存储器 806,它们经由总线 808 相互通信。计算机系统 800 还包括视频显示单元 810(例如,液晶显示(LCD)或者阴极射线管(CRT))。计算机系统 800 还包括字母数字输入设备 812(例如,键盘),光标控制设备 814(例如,鼠标),盘驱动单元 816,信号发生设备 818(例如,扬声器)和网络接口设备 820。

[0059] 盘驱动单元 816 包括机器可读介质 822,在其上存储了包含在这里描述的任何的一个或者多个方法或功能的一个或者多个指令集(例如,软件 824)。软件 824 也可以完全地或者至少部分地驻留在主存储器 804 内,和/或驻留在由计算机系统 800 执行它的执行期间的处理器 802 内,主存储器 804 和处理器 802 也构建机器可读介质。

[0060] 可以进一步将软件 824 经由网络接口设备 820 通过网络 826 发送或接收。

[0061] 虽然在示例实施例中示出机器可读介质 822 是单个介质,但是术语“机器可读介质”应当包括单个介质或者多个介质(例如,集中的或者分布的数据库,和/或相关联的线缆和服务器),其存储一个或者多个指令集。术语“机器可读介质”还应当包括能够存储,编码或者承载指令集用于通过机器执行的任何介质,该指令集令机器执行本发明的任何一个或者多个方法。术语“机器可读介质”应当相应地包括但是不限于固态存储器,光和磁介质,载波信号。

[0062] 因此,描述了用于管理使用模型框架的多个域架构的方法和系统。虽然参考具体实施例已经描述了本发明,将显而易见的是在不背离本发明的更广的精神和范围的情况下可以对这些实施例做出多种修改和变化。因此,该说明书和附图应当被视为解释的而不是限制的意义。

[0063] 在这里描述的一些实施例可以被用于解决一个或者多个技术问题。例如一些实施例可以便利更有效的资源管理和减少当添加或者修改一个架构域时重新部署系统的需要。

[0064] 提供了本公开的摘要以遵从 37 C.F.R. § 1.72(b),其要求允许读者快速地确定该技术公开的特征的摘要。理解该摘要不被用于解释或者限定权利要求的范围或者意义地来提交该摘要。另外,在前述详细地描述中,可以看出为了精简本公开的目的将多种特征在单个实施例中组合在一起。不将本公开的方法解释为反映意图:要求的实施例比在每一权利

要求中明确叙述的要求更多的特征。相反,如权利要求反映的,发明的主题取决于少于单个公开的实施例的全部特征。因此将权利要求结合在详细地描述中,位于它自己上的每一权利作为一个单独的实施例。

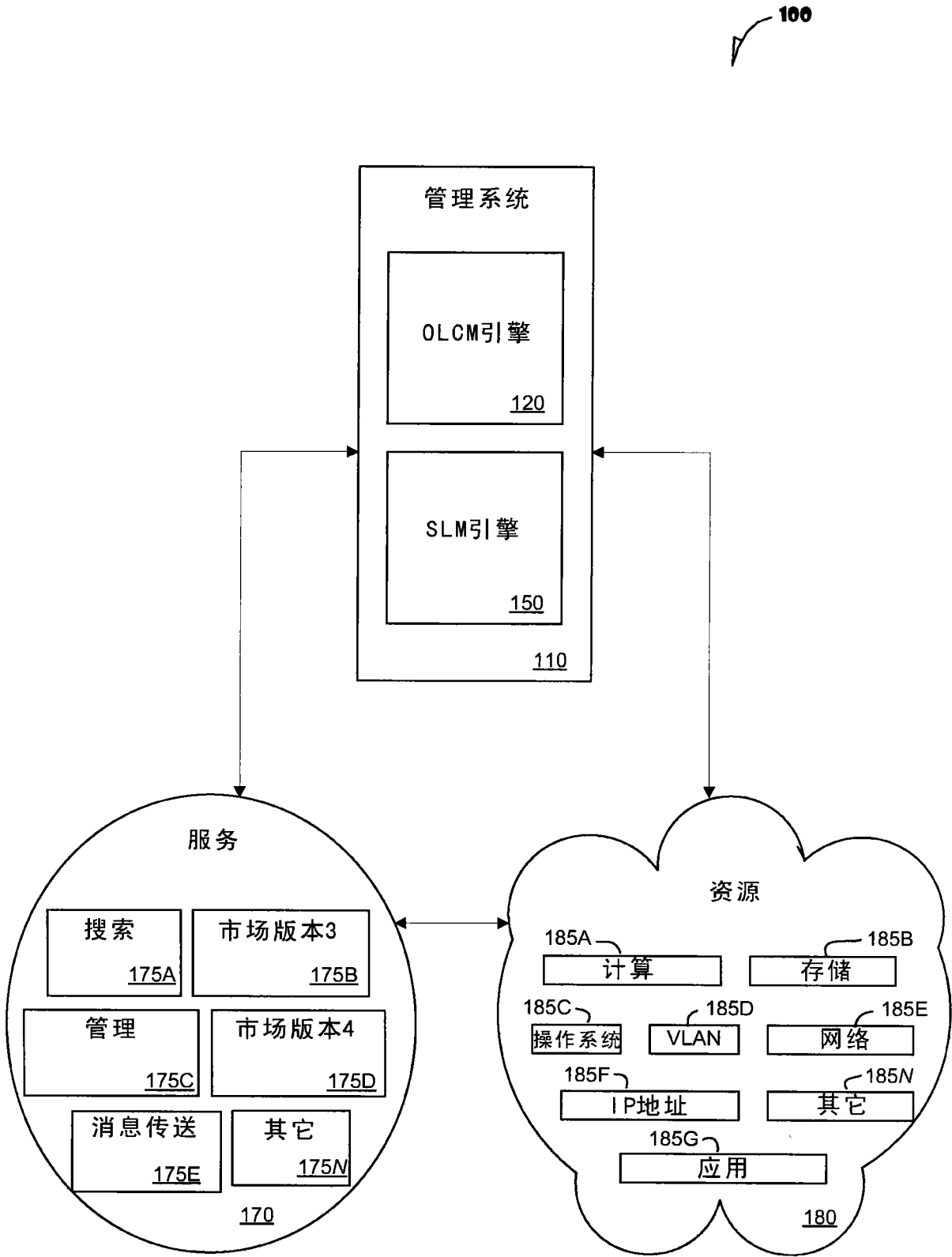


图 1

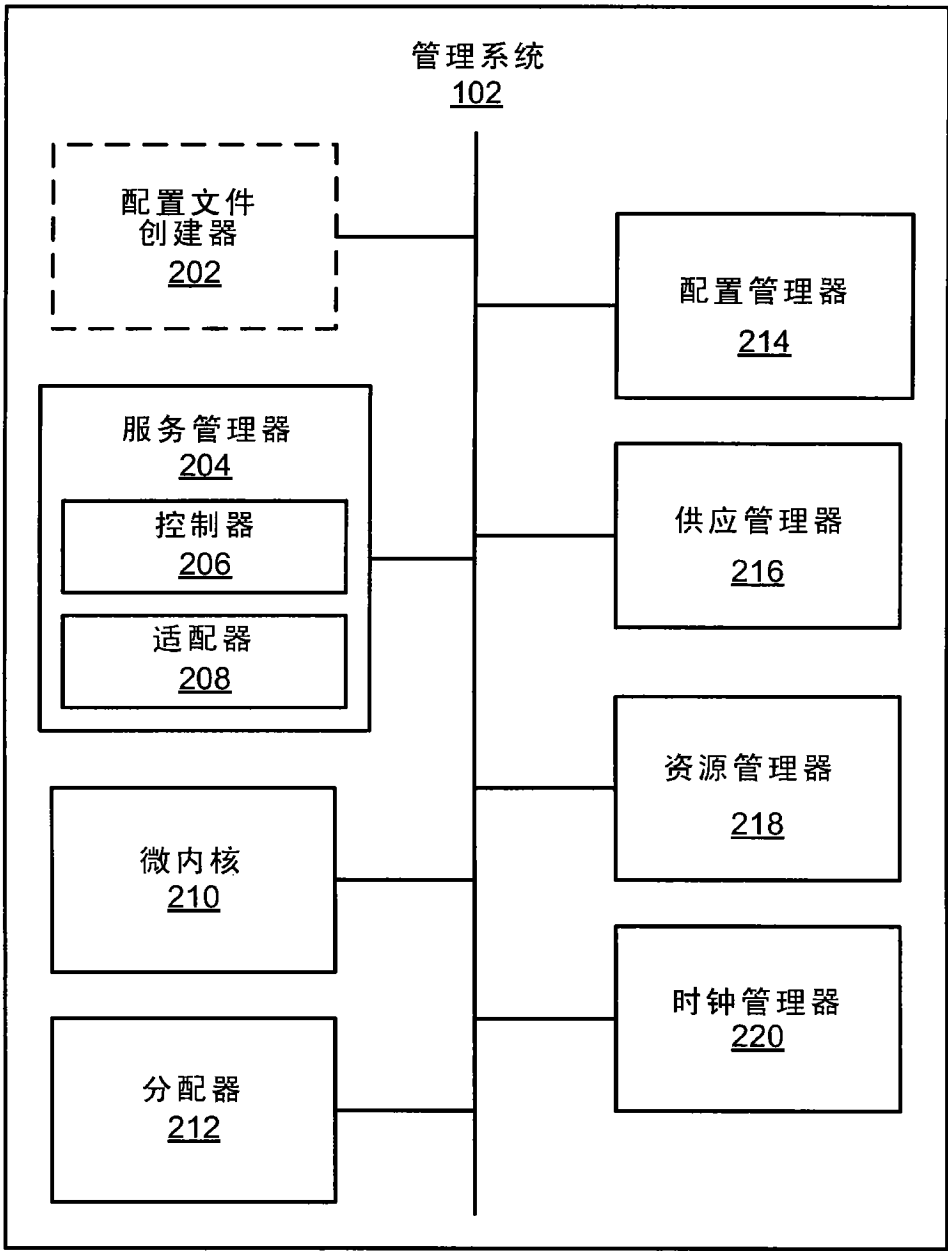


图 2

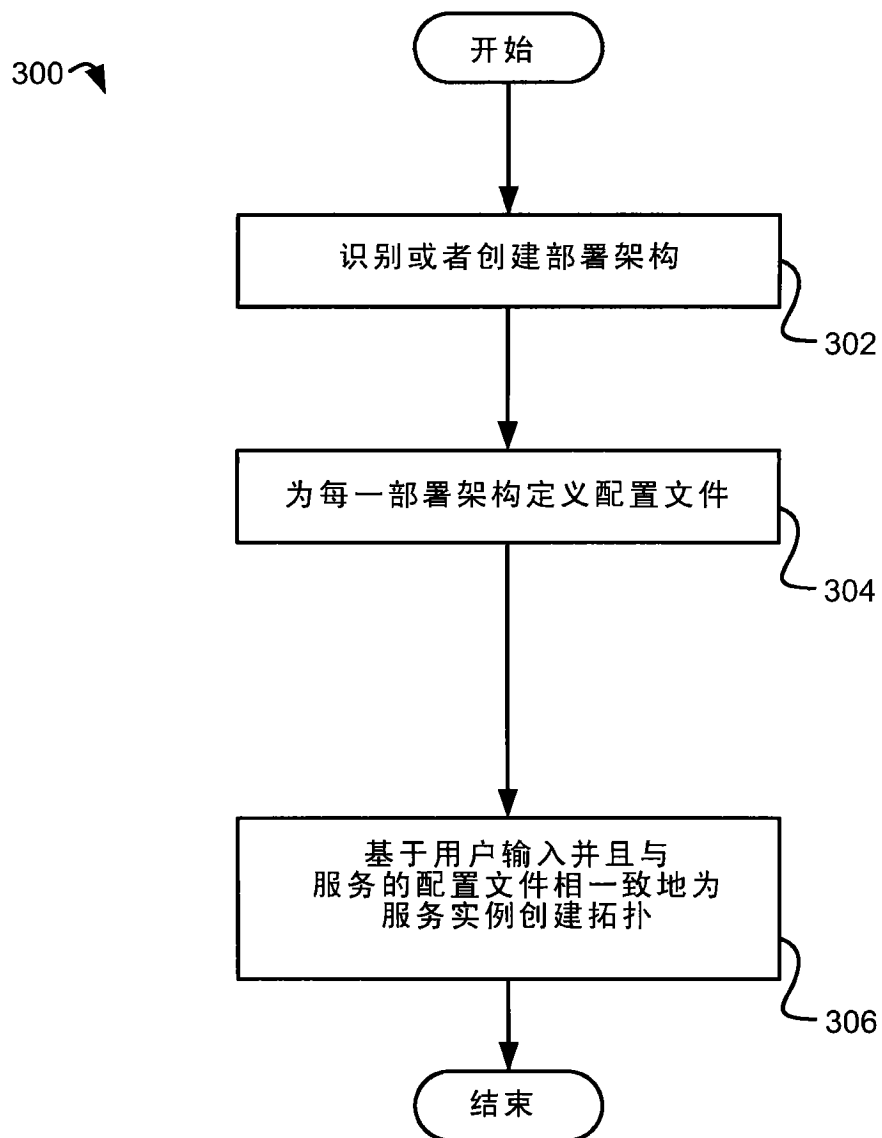


图 3

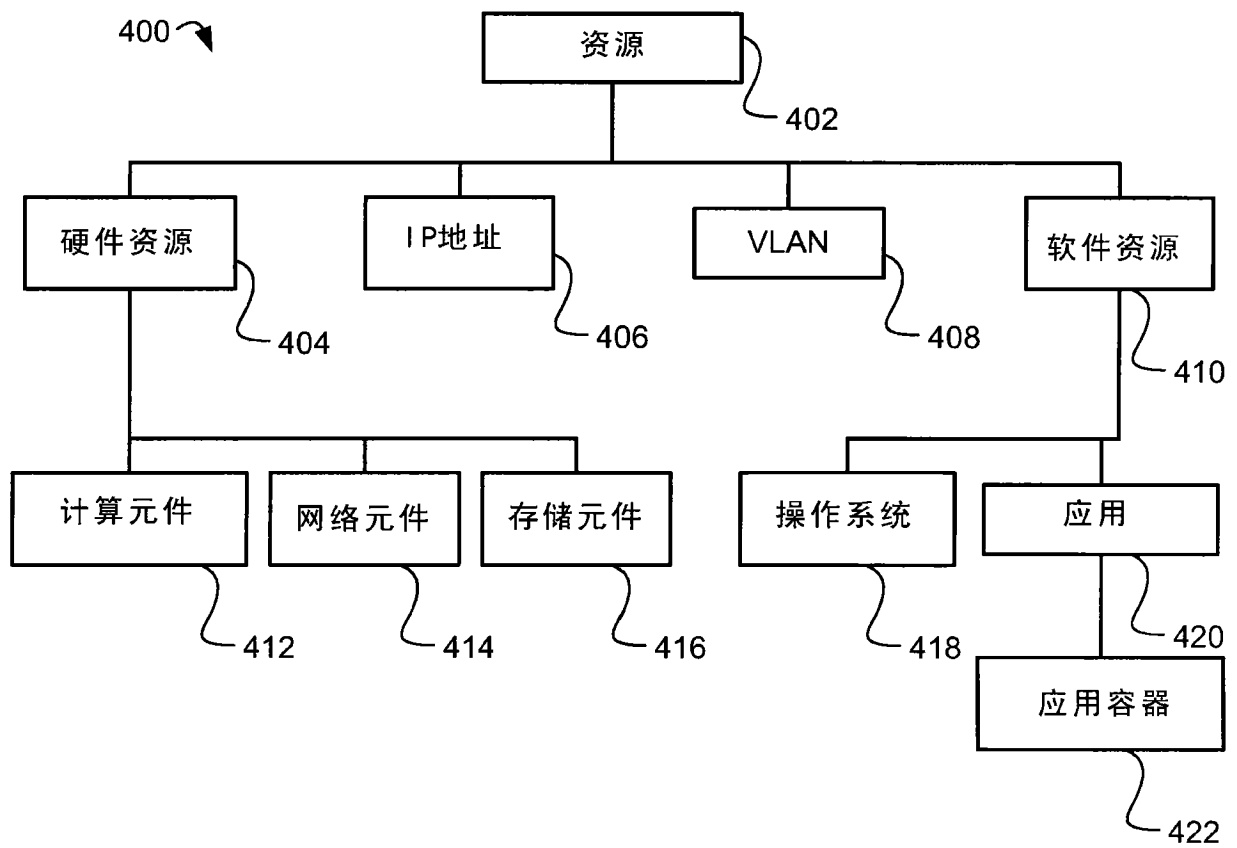


图 4

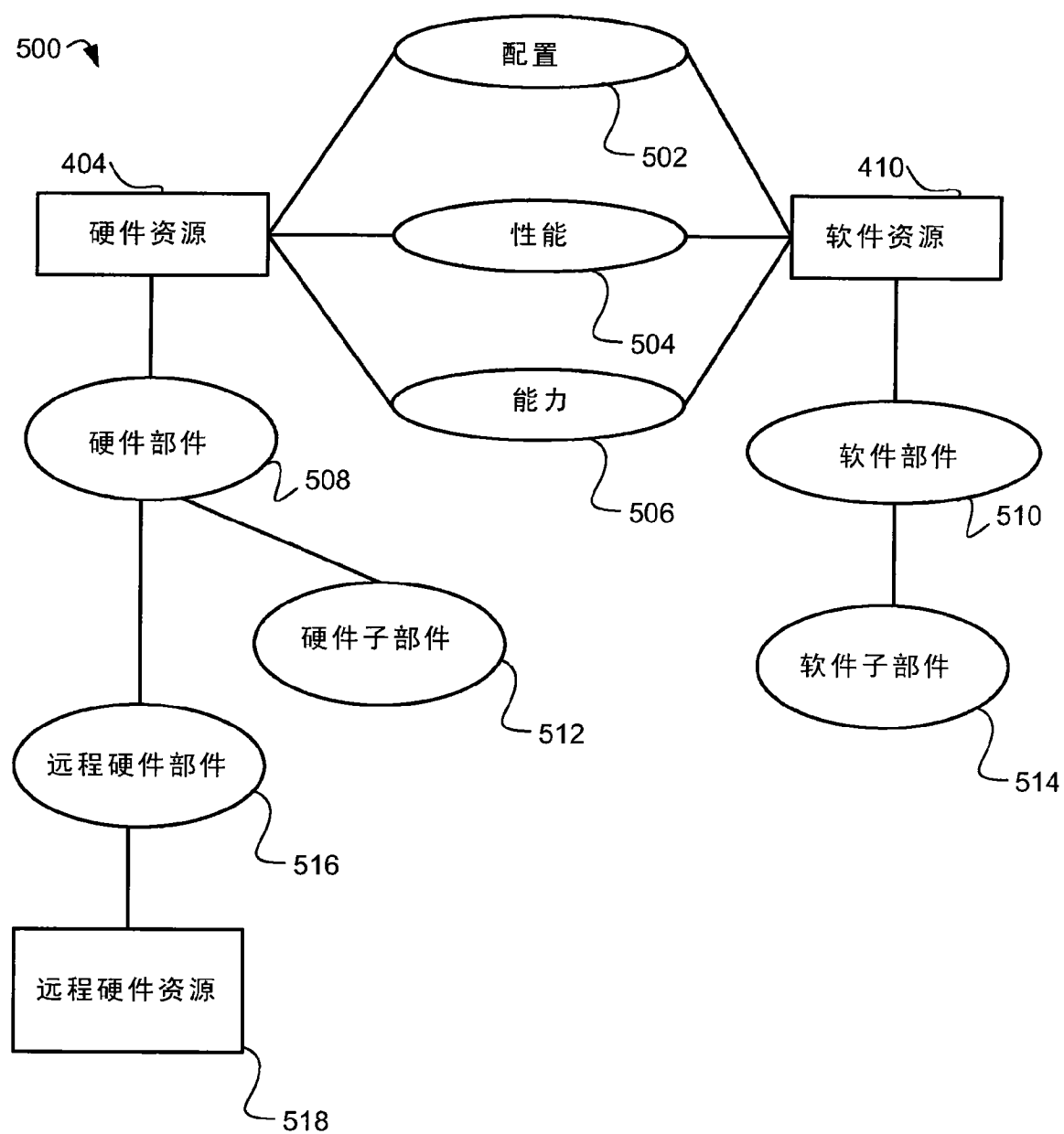


图 5

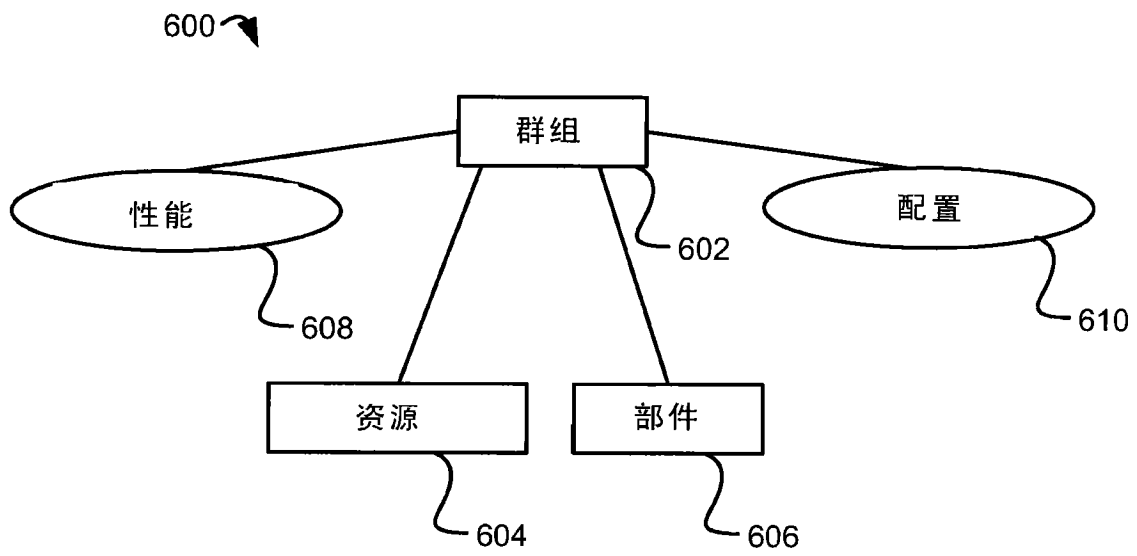


图 6

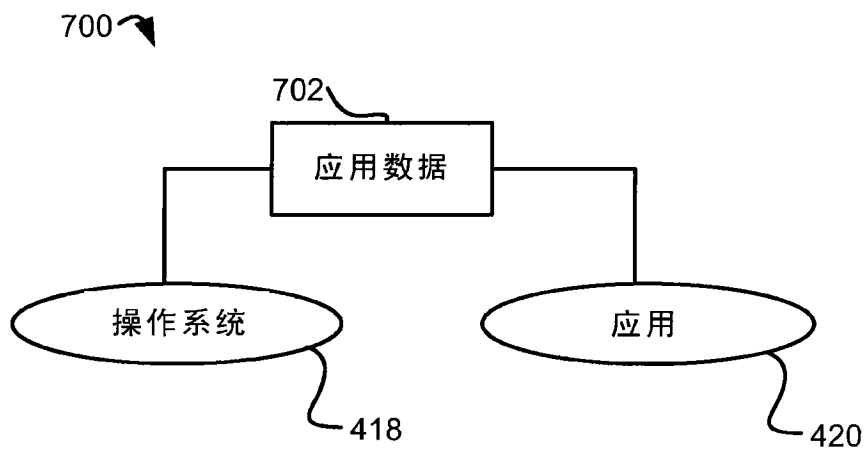


图 7

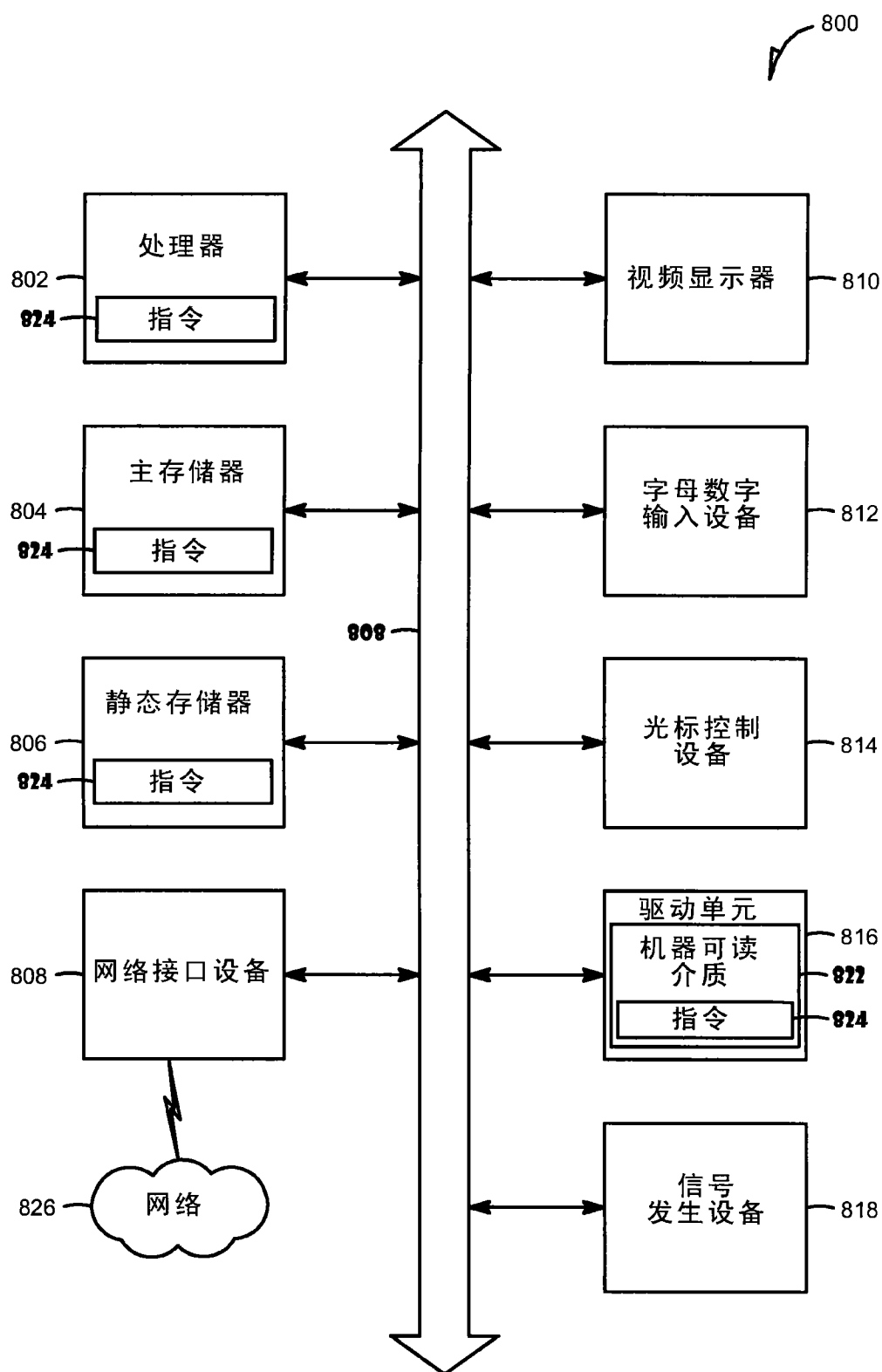


图 8

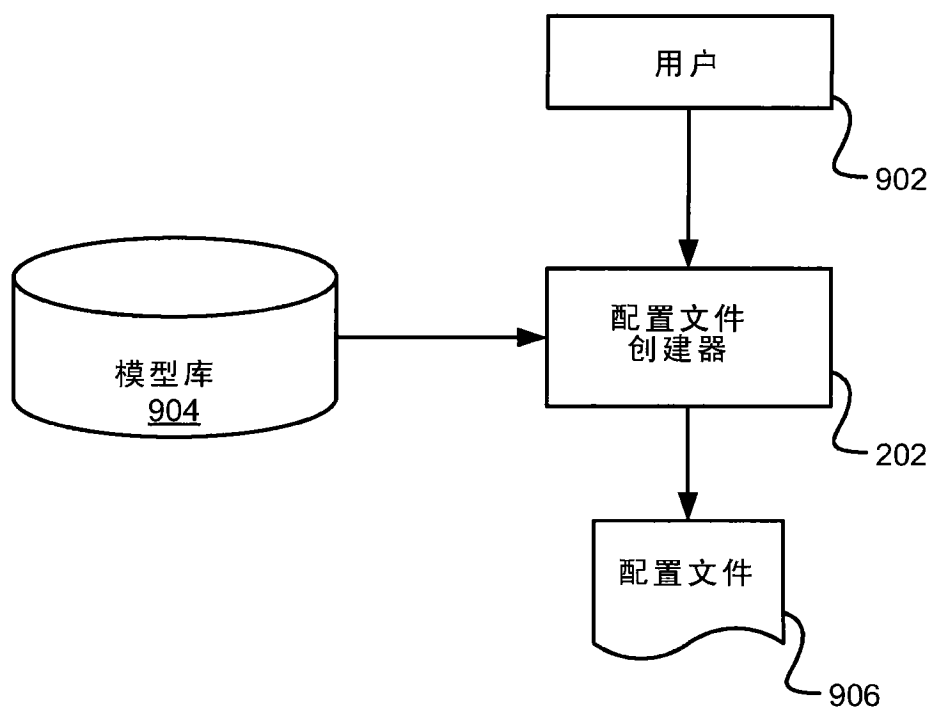


图 9

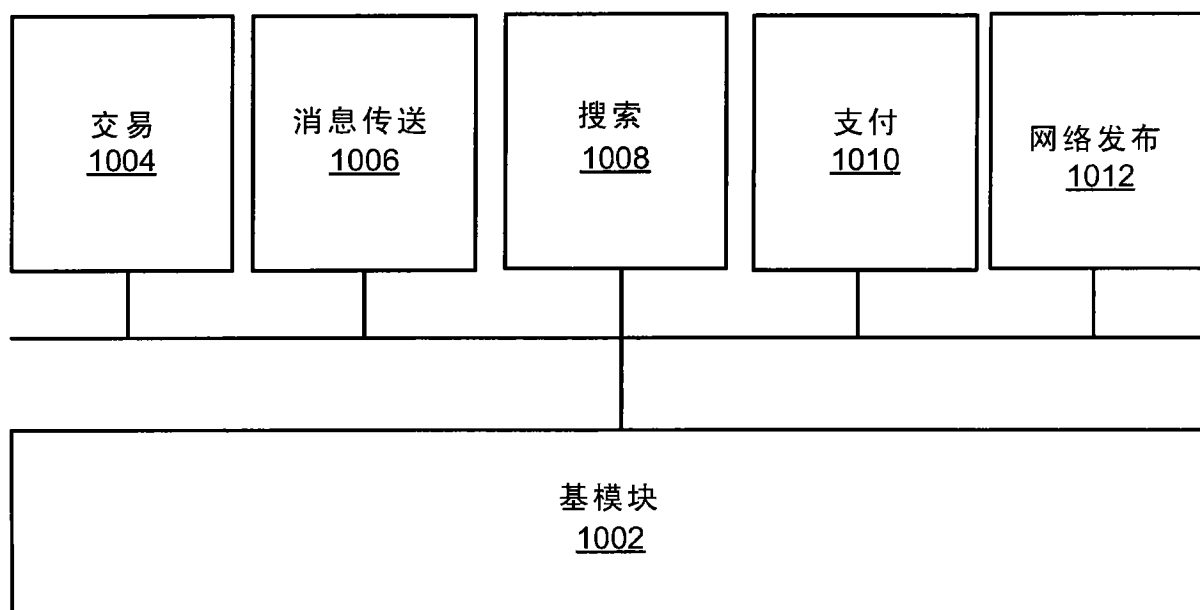


图 10