(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 114930293 A
(43)申请公布日 2022.08.19

(21)申请号 202080056503.4

(22)申请日 2020.06.12

(30)优先权数据
62/860,740 2019.06.12 US
62/864,476 2019.06.20 US

(85)PCT国际申请进入国家阶段日
2022.02.09

(86)PCT国际申请的申请数据
PCT/US2020/037598 2020.06.12

(87)PCT国际申请的公布数据
WO2020/252390 EN 2020.12.17

(71)申请人 SNYK有限公司
地址 英国伯克郡

(72)发明人 杰文·麦克唐纳 詹姆斯·鲍斯 多梅尼克·罗萨蒂

(74)专利代理机构 北京银龙知识产权代理有限公司 11243
专利代理师 龚伟 李鹤松

(51)Int.Cl.
G06F 9/50(2006.01)
H04L 12/02(2006.01)
G06F 9/44(2018.01)
G06F 11/30(2006.01)
G06F 11/34(2006.01)

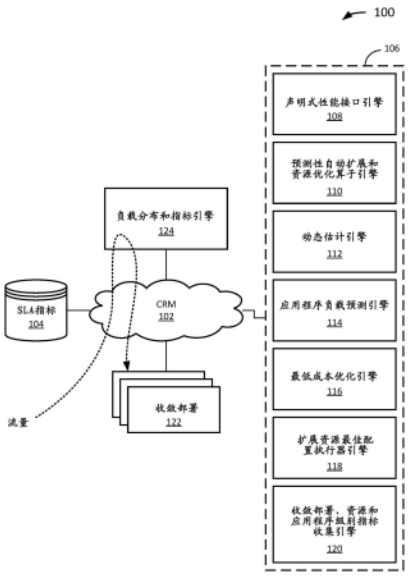
权利要求书3页 说明书12页 附图5页

(54)发明名称

预测性自动扩展和资源优化

(57)摘要

用于软件部署的预测性自动扩展和资源优化的技术。在一个实现方式中，用户声明性能目的，并且使用应用程序行为和负载配置文件的机器学习来确定满足声明的性能目的的最低成本资源配置。在一个实施方式中，监控收敛部署并且提供相关的反馈以随着时间改进预测、行为建模和资源估计。



CN 114930293 A

1.一种系统,其包括:

声明式性能接口引擎,其被配置为:

接收服务级别协议(SLA)指标;

将所述SLA指标转换为声明式性能数据结构,该数据结构表示软件部署的声明的性能目的;

监控部署的软件的性能指示器;

收敛部署、资源和应用程序级别指标收集引擎,其被配置为监控部署的软件的资源使用;

动态估计引擎,其被配置为生成应用程序行为模型,该应用程序行为模型基于随负载而变的性能指示器和资源使用;

应用程序负载预测引擎,其被配置为预测未来时间的负载;

最低成本优化引擎,其被配置为基于所述声明式性能数据结构、所述应用程序行为模型和预测的负载来生成最低成本优化参数;

预测性自动扩展和资源优化算子引擎,其被配置为向扩展资源最佳配置执行器引擎提供所述最低成本优化参数;

被配置为执行收敛部署的扩展资源最佳配置执行器引擎;

负载分配和指标引擎,其被配置为对到所述收敛部署的流量和来自所述收敛部署的流量中的一项或多项执行负载平衡,所述负载分配和指标引擎由所述扩展资源最佳配置执行器引擎根据所述最低成本优化参数来配置;

所述收敛部署、资源和应用程序级别指标收集引擎还被配置为:

监控与所述收敛部署相关联的资源;

向所述动态估计引擎提供与所述收敛部署相关联的反馈;

所述动态估计引擎还被配置为基于所述反馈生成更新的应用程序行为模型。

2.根据权利要求1所述的系统,其中,所述SLA指标包括服务级别指示器(SLI)指标和服务级别目的(SLO)指标中的一个或多个。

3.根据权利要求1所述的系统,其中,所述SLA指标由人类代理定义。

4.根据权利要求1所述的系统,其中,所述SLA指标由人工代理定义。

5.根据权利要求1所述的系统,其中,所述性能指示器包括请求计数和请求持续时间中的一项或多项。

6.根据权利要求1所述的系统,其中,所述资源使用包括内存、CPU功率、磁盘I/O和网络I/O中的一种或多种的使用。

7.根据权利要求1所述的系统,其中,所述应用程序负载预测引擎被配置为预测季节性负载、流行负载、突发负载和随机负载中的一种或多种。

8.根据权利要求1所述的系统,其中,所述应用程序负载预测引擎被配置为估计用于设置资源限制的预测模式。

9.根据权利要求1所述的系统,其中,所述动态估计引擎使用深度学习生成所述应用程序行为模型。

10.根据权利要求1所述的系统,其还包括被配置为根据所声明的性能目的确定测量误差的预测性自动扩展和资源优化算子引擎。

11.一种方法,其包括:

接收服务级别协议(SLA)指标;

将所述SLA指标转换为声明式性能数据结构,该数据结构表示软件部署的声明的性能目的;

监控部署的软件的性能指示器和资源使用;

基于随负载而变的性能指示器和资源使用来生成应用程序行为模型;

预测未来时间的负载;

基于所述声明式性能数据结构、所述应用程序行为模型和预测的负载来生成最低成本优化参数;

执行收敛部署;

根据所述最低成本优化参数对到所述收敛部署的流量和来自所述收敛部署的流量中的一项或多项执行负载平衡;

监控与所述收敛部署相关联的资源;

提供与所述收敛部署相关联的反馈;

基于所述反馈生成更新的应用程序行为模型。

12.根据权利要求11所述的方法,其中,所述SLA指标包括服务级别指示器(SLI)指标和服务级别目的(SLO)指标中的一个或多个。

13.根据权利要求11所述的方法,其中,所述SLA指标由人工代理定义。

14.根据权利要求11所述的方法,其中,所述性能指示器包括请求计数和请求持续时间中的一项或多项。

15.根据权利要求11所述的方法,其中,所述资源使用包括内存、CPU功率、磁盘I/O和网络I/O中的一种或多种的使用。

16.根据权利要求11所述的方法,其中,预测的负载包括季节性负载、流行负载、突发负载和随机负载中的一种或多种。

17.根据权利要求11所述的方法,其还包括估计用于设置资源限制的预测模式。

18.根据权利要求11所述的方法,其中,所述应用程序行为模型是使用深度学习生成的。

19.根据权利要求11所述的方法,其还包括根据所声明的性能目的确定测量误差。

20.一种系统,其包括:

用于接收服务级别协议(SLA)指标的装置;

用于将所述SLA指标转换为声明式性能数据结构的装置,所述数据结构表示软件部署的声明的性能目的;

用于监控部署的软件的性能指示器和资源使用的装置;

用于基于随负载而变的性能指示器和资源使用来生成应用程序行为模型的装置;

用于预测未来时间的负载的装置;

用于基于所述声明式性能数据结构、所述应用程序行为模型和预测的负载来生成最低成本优化参数的装置;

用于执行收敛部署的装置;

用于根据所述最低成本优化参数对到所述收敛部署的流量和来自所述收敛部署的流

量中的一项或多项执行负载平衡的装置；

用于监控与所述收敛部署相关联的资源的装置；

用于提供与所述收敛部署相关联的反馈的装置；

用于基于所述反馈生成更新的应用程序行为模型的装置。

预测性自动扩展和资源优化

背景技术

[0001] 软件操作的容量规划和成本优化是正在进行的研究和开发的领域。过度供应会导致资源浪费和额外的成本，但行业标准平均为过度供应80-93％。供应不足会导致性能下降和违反SLA。研究表明，Web应用程序的性能下降可能会导致流失增加高达75％。“云服务可用性的初步结果……显示平均每年有7.738小时不可用或99.91％的可用性……基于行业接受的每小时成本，这些故障的成本总计接近2.85亿美元。”Cérin等人的当前云方案的停机时间统计(更新版本—2014年3月)。

[0002] 由于资源利用率和应用程序负载的动态特性，手动确定容量实际上总是错误的。根据定义，反应性自动扩展(autoscaling)无法提前满足负载。基于阈值的自动扩展需要大量工作，并且即使在使用自定义应用程序级别指标时也无法与定义的服务级别目的一致。最多80％的利用率阈值会导致20％的容量不足。研究表明，基于阈值的自动扩展器无法适应变化的工作负载。

[0003] 解决与容量规划和成本优化相关联的这些和其他缺陷的技术是期望的。

发明内容

[0004] 公开了一种用于软件资源配置(resourcing)和扩展的成本和性能管理方案。在特定的实现方式中，系统在软件部署编排平台(例如Kubernetes)上运行，该软件部署编排平台公开应用程序和资源指标并提供标准的扩展和资源配置机制。用户声明性能目的，并且系统学习应用程序行为和负载配置文件来确定最低成本资源配置以满足声明的性能目的。

附图说明

[0005] 图1描绘了预测性自动扩展和资源优化系统的例子的图表。

[0006] 图2描绘了将按照反应性推荐引擎的推荐的资源供应与按照预测的推荐引擎的推荐的资源供应进行比较的图表。

[0007] 图3是预测的总负载对比实际的总负载的图表和相关联的代码显示的图表。

[0008] 图4描绘了预测性自动扩展和资源优化的方法的例子的流程图。

[0009] 图5描绘了与机器学习过程结合生成预测性自动扩展和资源优化结果的例子的流程图。

[0010] 图6描绘了用于生成最低成本优化参数的系统的例子的图表。

具体实施方式

[0011] 图1描绘了预测性自动扩展和资源优化系统的例子的图表100。如在本文中所使用的，资源可以被表征为中央处理单元(CPU)、内存、网络输入/输出(I/O)、磁盘I/O、图形处理单元(GPU)和/或其他适用资源。图表100包括计算机可读介质(CRM)102、与CRM 102耦合的服务级别协议(SLA)指标数据存储区(datastore)104、与CRM 102耦合的用于软件编排平台的前馈控制系统106、与CRM 102耦合的收敛部署(convergent deployment)122以及与CRM

102耦合的负载分布和指标引擎124。用于软件编排平台的前馈控制系统106包括声明式性能接口引擎108、预测性自动扩展和资源优化算子引擎110、动态估计引擎112、应用程序负载预测引擎114、最低成本优化引擎116、扩展资源最佳配置执行器引擎118以及收敛部署、资源和应用程序级别指标收集引擎120。预测性自动扩展和资源优化系统可以完全被实现；被实现为分阶段集成(例如，集成到Kubernetes集群中)，其中客户对是否实时进行更改以及允许进行多少更改进行控制；或使用平台数据样本进行实现以提供成本节约和/或性能改进报告。

[0012] CRM 102旨在表示计算机系统或计算机系统网络。如本文的使用，“计算机系统”可以包括或被实现为用于执行本文描述的功能的专用计算机系统。通常，计算机系统包括处理器、内存、非易失性存储器和接口。典型的计算机系统通常至少包括处理器、内存和将内存与处理器耦合的设备(例如，总线)。处理器例如可以是通用中央处理单元(CPU)(例如微处理器)或专用处理器(例如微控制器)。

[0013] 例如但非限制性地，计算机系统的存储器包括随机存取存储器(RAM)，例如动态RAM(DRAM)和静态RAM(SRAM)。内存可以是本地的、远程的或分布式的。非易失性存储器通常是磁式软盘或硬盘、磁式光盘、光盘、只读存储器(ROM)(例如CD-ROM、EPROM或EEPROM)、磁卡或光卡或用于大量数据的另一种存储形式。在软件执行期间，该数据中的一些通常通过与非易失性存储器耦合的总线通过直接内存访问过程写入内存中。非易失性存储器可以是本地的、远程的或分布式的，但它是可选的，因为可以使用内存中可用的所有适用数据创建系统。

[0014] 计算机系统中的软件通常存储在非易失性存储器中。实际上，对于大型程序，甚至可能无法将整个程序存储在内存中。为使软件运行，必要时，将其移动到适合处理的计算机可读位置，并且在本文中，出于说明的目的，该位置被称为内存。即使将软件移动到内存中进行执行，处理器通常也会利用硬件寄存器来存储与软件相关联的值，以及理想情况下用于加速执行的本地缓存。如本文的使用，当软件程序被称为“在计算机可读存储介质中实现”时，假定该软件程序存储在适用的已知或方便的位置(从非易失性存储器到硬件寄存器)。当与程序相关联的至少一个值被存储在处理器可读的寄存器中时，该处理器被认为“被配置为执行该程序”。

[0015] 在一个操作例子中，计算机系统可以由操作系统软件控制，操作系统软件是包括文件管理系统的软件程序，例如磁盘操作系统。具有相关联的文件管理系统软件的操作系统软件的一个例子是来自华盛顿州雷德蒙德的微软公司的被称为Windows的操作系统家族及其相关联的文件管理系统。具有相关联的文件管理系统软件的操作系统软件的另一个例子是Linux操作系统及其相关联的文件管理系统。文件管理系统通常存储在非易失性存储器中，并使处理器执行操作系统所需的各种动作以输入和输出数据并将数据存储于内存中，包括将文件存储于非易失性存储器上。

[0016] 计算机系统的总线可以将处理器耦合至接口。接口有助于设备和计算机系统的耦合。接口可以用于输入和/或输出(I/O)设备、调制解调器或网络。例如但非限制性地，I/O设备可以包括键盘、鼠标或其他定点设备、磁盘驱动器、打印机、扫描仪和其他I/O设备，包括显示设备。例如但非限制性地，显示设备可以包括阴极射线管(CRT)、液晶显示器(LCD)或一些其他适用的已知或方便的显示设备。例如但非限制性地，调制解调器可以包括模拟调制

解调器、IDSN调制解调器、电缆调制解调器和其他调制解调器。例如但非限制性地，网络接口可以包括令牌环接口、卫星传输接口(例如“直接PC”)或用于将第一计算机系统耦合至第二计算机系统的其他网络接口。接口可以被认为是设备或计算机系统的一部分。

[0017] 计算机系统可以与基于云的计算系统兼容或作为基于云的计算系统的一部分或通过基于云的计算系统来实现。如本文的使用，基于云的计算系统是向客户端设备提供虚拟化的计算资源、软件和/或信息的系统。计算资源、软件和/或信息可以通过维护边缘设备可以通过诸如网络的通信接口访问的集中式服务和资源来虚拟化。“云”可以是营销术语，就本文而言，可以包括本文描述的任何网络。基于云的计算系统可以涉及服务订阅或使用效用定价模型。用户可以通过Web浏览器或位于其客户端设备上的其他容器应用程序(container application)访问基于云的计算系统的协议。

[0018] 计算机系统可以作为引擎、作为引擎的一部分或通过多个引擎来实现。如本文的使用，引擎至少包括两个组件：1)专用或共享的处理器或其一部分；2)处理器执行的硬件、固件和/或软件模块。一个或多个处理器的一部分可以包括少于包括任何指定的一个或多个处理器的所有硬件的硬件的一些部分，例如寄存器的子集、专用于多线程处理器的一个或多个线程的处理器的部分、处理器完全或部分专用于执行引擎的部分功能的时间切片(time slice)等。这样，第一引擎和第二引擎可以具有一个或多个专用处理器，或者第一引擎和第二引擎可以彼此或与其他引擎共享一个或多个处理器。根据实现方式特定的考虑或其他考虑，引擎可以是集中式的，或者其功能是分布式的。引擎可以包括包含在计算机可读介质中以供处理器执行的硬件、固件或软件。处理器使用实现的数据结构和方法将数据转换为新数据，例如参照本文的附图描述的那样。

[0019] 本文描述的引擎或者可以实现本文描述的系统和设备的引擎可以是基于云的引擎。如本文的使用，基于云的引擎是可以使用基于云的计算系统来运行应用程序和/或功能的引擎。应用程序和/或功能的全部或部分可以分布在多个计算设备上，并且不必仅限于一个计算设备。在一些实施方式中，基于云的引擎可以执行最终用户通过网络浏览器或容器应用程序访问的功能和/或模块，而无需将该功能和/或模块在本地安装在最终用户的计算设备上。

[0020] 如本文的使用，数据存储区旨在包括具有任何适用的数据组织的存储库，包括表格、逗号分隔值(CSV)文件、传统数据库(例如SQL)或其他适用的已知或方便的组织格式。例如，数据存储区可以被实现为软件，该软件包含在通用或专用机器上的物理计算机可读介质中、在固件中、在硬件中、在它们的组合中或在适用的已知或方便的设备或系统中。尽管与数据存储区相关联的组件的物理位置和其他特征对于理解本文描述的技术并不至关重要，但是与数据存储区相关联的组件(例如数据库接口)可以被视为数据存储区的“一部分”、某些其他系统组件的一部分或它们的组合。

[0021] 数据存储区可以包括数据结构。如本文的使用，数据结构与在计算机中存储和组织数据的特定方式相关联，以便可以在给定的语境内有效地使用它。数据结构通常基于计算机在其内存中的任何位置获取和存储由地址(即，可以自身存储在内存中并由程序操纵的位串)指定的数据的能力。因此，一些数据结构基于用算术运算计算数据项的地址；而其他数据结构基于在结构本身内存储数据项的地址。许多数据结构使用有时以非平凡的方式结合的这两种原则。数据结构的实现通常需要编写创建和操纵该结构的实例的一组过程。

本文描述的数据存储区可以是基于云的数据存储区。基于云的数据存储区是与基于云的计算系统和引擎兼容的数据存储区。

[0022] 假设CRM包括网络，该网络可以是适用的通信网络，例如互联网或基础设施网络。本文使用的术语“互联网”是指使用某些协议(例如TCP/IP协议)以及可能的其他协议(例如用于构成万维网(“网络”)的超文本标记语言(HTML)文档的超文本传输协议(HTTP))的网络中的某个网络。更一般地，网络可以包括例如广域网(WAN)、城域网(MAN)、校园网(CAN)或局域网(LAN)，但网络至少在理论上可以是适用的大小或以某种其他方式表征(仅举几个替代方案，例如，个人区域网络(PAN)或家庭区域网络(HAN))。网络可以包括企业专用网络和虚拟专用网络(统称为专用网络)。顾名思义，专用网络受单个实体的控制。专用网络可以包括总部和可选的区域办事处(统称为办事处)。许多办事处允许远程用户通过一些其他网络(例如互联网)连接到专用网络办事处。

[0023] 再次参照图1的例子，SLA指标数据存储区104旨在表示包括数据结构的数据存储区，该数据结构表示用于软件部署的声明的性能目的。SLA指标数据存储区104可以包括服务级别指示器(SLI)数据结构。在信息技术中，SLI是对服务提供者向客户提供的服务级别的度量。SLI构成了服务级别目的(SLO)的基础，而后者又构成了SLA的基础。SLA指标数据存储区104可以替代地或附加地包括SLO数据结构。因此，如本文的使用，SLI和SLO中的一个或两个可以被视为SLA指标。尽管出于说明性目的假定将粒度(granular)SLI或SLO转换为声明式性能目标，但SLI和/或SLO的组合可以形成为SLA数据结构，存储于SLA指标数据存储区104中，并转换为聚集的声明式性能目标。在特定的实现方式中，声明的性能目的代表至少部分负责软件部署的服务消费者。服务消费者可以包括构建、提供或托管需要包含一项或多项服务(例如，计算资源)以实现所需的功能和操作的软件产品的实体。例如，服务消费者可以包括公司、组织、机构、企业、团体、个人或一些其他适用的实体或实体组。

[0024] 用于软件编排平台的前馈控制系统106旨在表示包括引擎和数据存储区的系统，该引擎和数据存储区用于预先主动扩展以提前为预测的负载做准备的，从而减轻软件部署的供应延迟。应用程序和应用程序资源的预先主动扩展能够实现资源的供应以满足未来的负载。在各种实施方式中，用于软件编排平台的前馈控制系统106通过减少资源消耗来降低运行应用程序的成本；通过不断地为应用程序进行资源配置和扩展来提高应用程序的性能，以便它们能够满足当前负载和预测负载的性能目的；通过使资源配置和扩展机制退出并满足它来让用户对他们设定的SLO充满信心；和/或减少为了对应用程序正确地进行资源配置和扩展而分析应用程序所涉及的手动时间和精力。在特定的实现方式中，用于软件编排平台的前馈控制系统106以高达90％的准确度预测随机和有规律的工作负载；抢占式的资源配置导致SLA违规平均减少十分之九。应用程序的抢占式的资源配置意味着供应不足至少得到改善，并且在理想情况下被消除。在特定的实现方式中，用于软件编排平台的前馈控制系统106可以从做出的决策中学习，以便改进预测、建模、资源估计和其他适用的决策。

[0025] 声明式性能接口引擎108旨在表示与控制系统类型技术(与用于软件编排平台的前馈控制系统106的其他引擎结合描述的)协调使用以消耗来自SLA指标数据存储区104的与软件部署相关联的目标性能指标并向控制系统类型技术提供声明式性能目标以设计启动程序来实现该目标的引擎。在特定的实现方式中，声明式性能接口引擎108限定其声明可被表征为“声明式性能”并且其定义可被表征为“声明式性能目的”或“声明式性能目标”的

参考或目标。声明式性能接口引擎108使服务提供者(或服务消费者)的人类或人工代理能够定义SLA指标(例如SLI或SLO),以存储于SLA指标数据存储区104中作为用于软件编排平台的前馈控制系统106为应用程序进行资源配置的目标。通过声明性能,人类代理无需每次软件或应用程序负载发生变化时努力手动地分析他们的应用程序、配置他们的应用程序并且为他们的应用程序进行资源配置;并且人工代理不必是人工智能。有利地,声明式性能接口引擎108允许服务提供者依赖用于提供给服务消费者的SLA的声明的SLO(或从一个或多个声明的SLI导出的SLO)。在特定的实现方式中,实现本文描述的声明式性能接口引擎108和控制系统类型技术导致SLA合规性平均增加70%。

[0026] 预测性自动扩展和资源优化算子引擎110旨在表示响应于预测的资源需求而自动扩展而不过度供应的引擎。资源优化旨在意味着供应满足声明的性能目的所需的最少资源。在特定的实现方式中,预测性自动扩展和资源优化算子引擎110自动做出改变,推荐更具成本效益的SLO,并发送关于潜在性能下降的警报。在特定的实现方式中,预测性自动扩展和资源优化算子引擎110通过使用对趋势和季节性敏感的深度学习方法对季节性和随机的应用程序负载和资源签名是稳健的,并且被训练为对随机突发的领先指示器敏感。在特定的实现方式中,预测性自动扩展和资源优化算子引擎110易于安装,从而支持可互换的指标收集和负载平衡器;能够在云端或本地(on-prem)运行;并且可以在短短5分钟内做出推荐(与用于软件编排平台的前馈控制系统106的引擎和数据存储区协调)。

[0027] 在特定的实现方式中,预测性自动扩展和资源优化算子引擎110不干扰Kubernetes调度器。Kubernetes是一个由云原生计算基金会维护的用于使应用程序部署、扩展和管理自动化的开源容器编排系统。它旨在提供一个“用于跨主机集群使应用程序容器的部署、扩展和操作自动化的平台”。它利用一系列容器工具,包括Docker。许多云服务提供基于Kubernetes的平台或基础设施作为服务(PaaS或IaaS),Kubernetes可以在该服务上被部署为提供平台的服务。许多供应商还提供自己品牌的Kubernetes发行版。Brian Grant、Tim Hockin和Clayton Colman的、最后更新于2017年4月20日的题为“The Kubernetes Architectural Roadmap”的文献通过引用并入本文。

[0028] 在特定的实现方式中,预测性自动扩展和资源优化算子引擎110与水平自动扩展器和集群自动扩展器一起很好地工作。例如,Kubernetes水平pod自动扩展器会基于观察到的CPU利用率而自动扩展复制控制器、部署或复制集(replicaset)中的pod数量。作为另一个例子,Oracle云平台允许服务器实例通过定义基于CPU和/或内存利用率的自动扩展规则来自动向内或向外扩展集群,以确定何时添加或删除节点。

[0029] 在特定的实现方式中,预测性自动扩展和资源优化算子引擎110将以下输出接收作为输入:1)来自收敛部署、资源和应用程序级别指标收集引擎120的测量的输出和2)来自应用程序负载预测引擎114的预测的输出。在特定的实现方式中,预测性自动扩展和资源优化算子引擎110根据性能目的确定测量误差,该测量误差被提供给动态估计引擎112。

[0030] 动态估计引擎112旨在表示估计在预测的和当前的负载下满足服务级别目的所需的最小资源量的引擎。在特定的实现方式中,动态估计引擎112在资源利用率方面对应用程序行为建模,并将性能建模为对负载的响应。通过对应用程序在负载下的响应和资源利用率进行建模,可以实现对垂直和水平自动扩展的估计,并基于建模来估计在特定负载下将使用多少资源以及对应的服务指示器(例如处理请求的时间);相应地设置部署请求。

[0031] 应用程序负载预测引擎114提供对未来时间的负载估计。根据实现方式特定的因素，未来时间可配置为短至未来一分钟或长达未来一小时。在特定的实现方式中，应用程序负载预测引擎114提供了91％准确的负载预测。有利地，通过提前预测传入的应用程序负载，可以在负载事件发生之前纵向扩展，甚至跨越各种各样的工作负载。例如，使用已被证明可以跨越各种各样的工作负载泛化(generalize)的深度学习方法，应用程序负载预测引擎114可以高度准确地(至少83％或高达95％以上)预测季节性的、流行的(trendy)、突发的和随机的负载。基于请求和对工作负载的了解，可以根据利用模式在部署上设置签名限制(例如，突发导致较高的限制，而稳定导致较低的限制)。

[0032] 最低成本优化引擎116使用当前负载、来自应用程序负载预测引擎114的预测负载、来自动态估计引擎112的应用程序行为模型以及来自声明式性能接口引擎108的声明的目的，来找到以适合声明的目的的性能运行建模的应用程序的最低成本，这是应用程序的副本(水平扩展)和资源(垂直扩展)之间的权衡。对性能目的的优化的关注会导致成本优化。有利地，在特定的实现方式中，这导致与没有这种关注的系统相比平均高达80％的成本节省。

[0033] 扩展资源最佳配置执行器引擎118旨在表示问题空间所特有的执行器。在特定的实现方式中，由于参数时间序列模型就其性质而言被调整到一种类型的时间序列配置文件，因此预测模型对不同的时间序列配置文件是稳健的。这些配置文件包括开关工作负载、突发工作负载、具有各种趋势的工作负载以及具有不同季节性成分(秒、分钟、小时等)的工作负载。这可以通过离线训练模型以针对许多不同的时间序列配置文件进行预测来实现；为此可以使用递归神经网络(recurrent neutral network)。然后在系统中部署离线模型，并且扩展资源最佳配置执行器引擎118可以被表征为是与一种类型的时间序列配置文件相关联的问题空间所特有的。

[0034] 扩展资源最佳配置执行器引擎118执行扩展资源的最佳配置，例如副本的数量、资源请求的大小和服务质量(由终止或压制应用程序资源使用的限制所定义)。通过了解负载下的应用程序，可以优化为了根据性能目的以最低成本满足预测需求而选择的副本和虚拟机(VM)实例类型(在网络和磁盘绑定应用程序的情况下)的数量，从而使满足预测的需求所需的资源最小化。

[0035] 在特定的实现方式中，扩展资源最佳配置执行器引擎118使用扩展资源(例如最大可分配资源)所特有的启发法和通过基于共识的推荐的振荡阻尼。扩展资源最佳配置执行器引擎118使应用程序作为一个收敛部署122而被执行。

[0036] 收敛部署、资源和应用级别指标收集引擎120旨在表示基于当前性能和预测负载来测量反馈和前馈(预测)的引擎。反馈可以以来自扩展资源最佳配置执行器引擎118或收敛部署122的系统输出的形式出现，或者以与通过CRM 102提供或在CRM 102上观察到的收敛部署122的相关收敛部署相关联的其他数据的形式出现。反馈和前馈被预测性自动扩展和资源优化算子引擎110用来调整推荐。在特定的实现方式中，收敛部署、资源和应用级别指标收集引擎120监控性能指示器和资源使用，包括诸如请求计数和请求持续时间的SLI，以及诸如内存、CPU、磁盘I/O和每个容器和pod的网络I/O的资源利用率指标。

[0037] 收敛部署122旨在表示以收敛配置执行应用程序的引擎。收敛配置是由扩展资源最佳配置执行器引擎118执行以结合预测性自动扩展和资源优化的配置。

[0038] 负载分布和指标引擎124旨在表示指定如何收集和配置应用程序负载指标以进行分布的引擎。在特定的实现方式中，负载分布和指标引擎124对去往(或来自)收敛部署122的流量执行负载平衡。负载平衡改进了跨越多个计算资源(例如计算机、计算机集群、网络链接、CPU或磁盘驱动器)的工作负载的分布。负载平衡旨在优化资源使用、使吞吐量最大化、使响应时间最小化并避免任何单个资源的过载。使用具有负载平衡的多个组件而不是单个组件可以通过冗余来提高可靠性和可用性。负载平衡通常涉及专用的软件或硬件，例如多层交换机或域名系统服务器进程。负载平衡与通道绑定的不同之处在于，负载平衡在网络套接字(network socket，OSI模型第4层)基础上划分网络接口之间的流量，而通道绑定意味着使用诸如最短路径桥接的协议在每个数据包的(OSI模型第3层)或在数据链路(OSI模型第2层)基础上的较低级别的物理接口之间划分流量。然而，通道绑定在本文中被视为负载平衡。在替代方案中，代理承担负载分布和指标功能，以代替可能被称为“负载平衡器”的功能。

[0039] 在本文中为了区分的目的，“负载分布和指标引擎”是比负载平衡器、代理或其他适用的特定应用程序负载分布和指标系统更通用的用于应用程序负载指标收集和分布配置的术语。实际上，在特定的实现方式中，使用应用程序负载指标收集和分布引擎而不使用负载平衡器。例如，负载分布和指标引擎124可以从应用程序代理(例如Envoy、L7代理和为大型调制解调器面向服务的架构所设计的通信总线)收集指标并对该应用程序代理进行配置。作为另一个例子，负载分布和指标引擎124可以利用其他负载系统，例如消息队列。一般而言，可以跨越不同的工作负载使用负载分布和指标引擎124(作为负载平衡满足的基于网络的工作负载的替代或补充)。

[0040] 负载分布和指标引擎124由扩展资源最佳配置执行器引擎118通知，以便以适合收敛部署122的方式平衡流量。收敛部署、资源和应用程序级别指标收集引擎120还可以从负载分布和指标引擎124收集数据。

[0041] 在操作的例子中，服务提供者(或服务消费者)的人类或人工代理使用声明式性能接口引擎108在SLA指标数据存储区104中存储SLA指标，例如SLI或SLO。在替代方案中，代理可以通过SLA指标数据存储区接口(未被示出)将SLA指标存储在SLA指标数据存储区104中。根据所存储的内容，SLA指标数据存储区104可以被称为SLI数据存储区、SLO数据存储区或SLA数据存储区。

[0042] 继续这个操作例子，声明式性能接口引擎108将数据结构转换为声明式性能目的，以供预测性自动扩展和资源优化算子引擎110消耗。当适用的数据变得从收敛部署、资源和应用程序级别指标收集引擎120可用时，动态估计引擎112在资源利用率方面对应用程序行为进行建模，并将性能建模为对负载的响应，并且应用程序负载预测引擎114提供对未来时间的负载估计。

[0043] 继续这个操作例子，最低成本优化引擎116使用来自应用程序负载预测引擎114的预测负载、来自动态估计引擎112的应用程序行为模型以及来自声明式性能接口引擎108的声明的目的，来找到以适合声明的目的的性能运行建模的应用程序的最低成本。预测性自动扩展和资源优化算子引擎110将最低成本优化参数提供给扩展资源最佳配置执行器引擎118，该扩展资源最佳配置执行器引擎118执行收敛部署122并根据最低成本优化参数配置负载分布和指标引擎124。在特定的实现方式中，配置负载分布和指标引擎124涉及到使所

供应的资源对负载分布和指标引擎124已知，这可以理所当然地发生。

[0044] 继续这个操作例子，收敛部署、资源和应用程序级别指标收集引擎120监控与收敛部署122相关联的通道和其他资源，它们可以被处理(以生成例如测量的输出)并作为反馈提供给预测性自动扩展和资源优化算子引擎110、动态估计引擎112和应用程序负载预测引擎114。反馈可用于提供初始数据集或随着时间的推移改进建模和推荐。

[0045] 图2描绘了将按照反应性推荐引擎的推荐的资源供应与按照预测的推荐引擎的推荐的资源供应进行比较的图表200。图表200包括资源消耗曲线202、预测性供应曲线204、反应性供应曲线206、性能下降区域208和浪费成本区域210。资源消耗曲线202旨在表示随时间(x轴)使用的资源量(y轴)。预测性供应曲线204旨在表示按照预测性推荐引擎的推荐供应的资源。反应性供应曲线206旨在表示按照作为预测性推荐引擎的替代的反应性推荐引擎的推荐分配的资源。

[0046] 如图表200所示，反应性供应曲线206的性能下降区域208大于预测性供应曲线204的性能下降区域208。实际上，预测性供应曲线204在测量的时间段内匹配或略微超过资源消耗曲线202，这意味着使用预测性推荐引擎的系统没有性能下降。可以注意到，当供应曲线小于资源消耗曲线202时，发生性能下降，这意味着已经发生供应不足。

[0047] 如图表200所示，反应性供应曲线206的浪费成本210大于预测性供应曲线204的浪费成本210。虽然预测性供应曲线204在图表200的大多数点处超过资源消耗曲线202，但是浪费成本的量明显小于与反应性供应曲线206相关联的浪费成本的量。反应性供应系统无法在对负载做出反应的5分钟之前实现正确的供应，因为虽然在5分钟之前有纵向扩展事件(例如，1到2分钟的反应时间)，但向下遵循曲线是困难的并且反应性算法会随着时间而降级。在特定的实现方式中，正确的供应(有供应保险(provisioning insurance))在负载后需要不到5分钟的时间。由于反应性供应系统在其接收指标、计算请求并启动这些之前无法实现正确的供应，因此反应性系统不可能在负载的一分钟内做出行动，这完全在特定的实现方式的能力范围内。有利地，在这个具有正确配置的预测性系统的特定的实现方式中，可以在负载或需要或消耗的资源之前x分钟内实现正确的供应，其中x是大于1分钟且小于1小时的可配置的前瞻时间。

[0048] 当供应曲线大于资源消耗曲线202加上供应保险时，会发生浪费成本。简单来说，在一分钟内资源消耗少于x％的浪费成本可以被称为供应保险，这在确保不会发生供应不足的许多情况下是可取的。供应保险可以被定义为资源值低于供应量的x％的可能性，或者是作为与平均值的2个标准偏差和/或峰均比(波峰因数)的95％的可能性。在特定的实现方式中，使用这两种启发法。资源值低于供应量的95％的可能性用于请求(例如，要请求多少资源)，而波峰因数用于确定限制(例如，允许应用程序在终止、压制或压缩资源使用之前消耗超出请求的资源数量)。软件编排平台中请求和限制之间的差异可以被称为服务质量(QoS)，该服务质量定义了是始终保证资源可用(即，请求和限制相同)还是允许软件在资源可用时根据需要超出其请求(即，限制高于请求)。

[0049] 图3是预测的总负载对比实际的总负载的图表302和相关联的代码显示304的图表300。预测的总负载对比实际的总负载的图表302具有时间戳秒的x轴和负载每秒计数的y轴。可以看出，预测的负载曲线总是以相对较小的幅度(供应保险幅度)超过请求计数曲线。相关联的代码显示304指示资源包括限制和请求，它们在前面的段落中进行了描述。

[0050] 图4描绘了预测性自动扩展和资源优化的方法的例子的流程图400。流程图400开始于将SLA指标数据结构转换为声明式性能目的的模块402。SLA指标数据结构可以存储于SLA指标数据存储区中，例如结合图1描述的SLA指标数据存储区104。声明式性能接口引擎(例如结合图1描述的声明式性能接口引擎108)可以将SLA指标数据结构转换为声明式性能对象。

[0051] 流程图400继续到估计对未来时间的负载预测的模块404。对于新部署，应该注意，负载预测可以具有有限的用途，因为它只相当于是基于有关部署的已知数据的猜测，而没有与资源利用率和部署后性能相关的反馈的好处。接收并处理这样的反馈通常需要几分钟，此时负载预测可以成为更具预测性的估计。因此，模块404可以根据不需要而被跳过，直到数据对做出准确的预测变得有用的时间为止。应用程序负载预测引擎(例如结合图1描述的应用程序负载预测引擎114)可以对未来时间估计负载预测。

[0052] 流程图400继续到使用负载预测和声明式性能目的来生成最低成本优化参数的模块406。如前一段所述，预测对于准确生成最低成本优化参数可能具有有限的用途。此外，尽管可以提供模型估计来代替响应于与部署相关联的反馈而生成的性能模型和应用程序行为模型，但是这样的模型的价值有限。在收到反馈后，模块406还可以使用性能模型和应用程序行为模型来生成最低成本优化参数。(参见下面结合模块416和418的描述。)最低成本优化引擎(例如结合图1描述的最低成本优化引擎116)可以使用负载预测、性能模型(如果适用)、应用程序行为模型(如果适用)和声明式性能目的来生成最低成本优化参数。

[0053] 流程图400继续到根据最低成本优化参数执行收敛部署的模块408。扩展资源最佳配置执行器引擎(例如结合图1描述的扩展资源最佳配置执行器引擎118)可以按照最低成本优化参数执行收敛部署。

[0054] 流程图400并行地继续到根据最低成本优化参数配置负载分布和指标引擎的模块410。尽管可以配置任何适用的模块以与另一个模块并行执行，但是模块408和410比较有可能并行地执行，因此相当明确地进行了图示。当然，可以重新布置模块以进行串行处理。扩展资源最佳配置执行器引擎(例如结合图1描述的扩展资源最佳配置执行器引擎118)可以根据最低成本优化参数配置负载分布和指标引擎。

[0055] 在模块408和410之后，流程图400继续到监控与收敛部署相关联的资源的模块412。收敛部署和资源指标收集引擎(例如结合图1描述的收敛部署、资源和应用程序级别指标收集引擎120)可以监控与收敛部署相关联的资源(包括通道)。

[0056] 流程图400继续到提供与收敛部署相关联的反馈的模块414。收敛部署和资源指标收集引擎(例如结合图1描述的收敛部署、资源和应用程序级别指标收集引擎120)可以提供与收敛部署相关联的反馈。

[0057] 流程图400返回到模块404并如前所述继续并且还(并行地)继续到在资源利用率方面对应用程序行为进行建模的模块416。在特定的实现方式中，应用程序行为的建模总共需要最多大约5分钟来对来自模块414的反馈进行接收、处理和执行机器学习。因此，在图4的例子的这种描述中，虽然可以使用“替代”模型，但模块416的引入不如模块404快。此外，流程图400可以在模块416完成之前多次循环通过其他模块。动态估计引擎(例如结合图1描述的动态估计引擎112)可以在资源利用率方面对应用程序行为进行建模。从模块416，流程图400返回到模块406并如前所述继续。

[0058] 流程图400还从模块414继续到将性能建模为对负载的响应的模块418。在特定的实现方式中，应用程序行为的建模总共需要最多大约5分钟来对来自模块414的反馈进行接收、处理和执行机器学习。因此，在图4的例子的这种描述中，虽然可以使用“替代”模型，但模块418的引入不如模块404快。此外，流程图400可以在模块418完成之前多次循环通过其他模块。动态估计引擎(例如结合图1描述的动态估计引擎112)可以将性能建模为对负载的响应。从模块418，流程图400返回到模块406并如前所述继续。

[0059] 可以注意到，可以并行处理模块404、416和418，但是出于实际目的，如果相对于来自第一循环的模型或预测没有对该模型或预测进行更新，则可以在从模块414到模块404、416和模块418的第二循环中跳过一个或多个模块。当然，也可以重新布置模块404、416和418以进行串行处理。还可以注意到，如果在没有SLA指标数据(将在进程中稍后提供)的情况下进行部署，则模块404、416和418可能会可想到地在模块402之前出现。最后，可以注意到，如果声明式性能目的改变(未被示出)，可以重复模块402。

[0060] 图5描绘了与机器学习过程结合生成预测性自动扩展和资源优化结果的例子的流程图500。流程图500以监控性能指示器和资源使用的模块502开始。服务提供者(或服务消费者)的人类或人工代理可以例如在审查收敛部署性能之后提供新的性能指示器。声明式性能接口引擎(例如结合图1描述的声明式性能接口引擎108)可以监控性能指示器。收敛部署和资源指标收集引擎(例如结合图1描述的收敛部署、资源和应用程序级别指标收集引擎120)可以监控资源使用。

[0061] 流程图500继续到预测应用程序负载和季节性的模块504。季节性可以结合用例来说明，在这个例子中，该用例例如是鞋企电子商务部署。成功的鞋企电子商务部署通常具有稳定的流量，在黑色星期五和假期期间具有一定的季节性，加上一些看似随机的高峰(例如，当新鞋发布时)。由于这些事件期间性能对收入的重要性，因此反应性自动扩展在这些时间是次佳的。系统工程师将手动地过度供应以满足SLO。如本文所述，预测性自动扩展和资源优化系统能够学习这些季节性并为这些事件提供正确的资源量(有供应保险)，而无需人工干预。此外，在这一年中，发布新鞋，系统工程师常常对发布期间的大量负载毫无准备。当他们反应性地纵向扩展以满足需求时，这双鞋在eBay上以10倍的价格出售。如本文所述，预测性自动扩展和资源优化系统能够预测这些看似随机的流量峰值并相应地扩展，从而满足SLO，并实现最大的收入和客户满意度。有利地，金钱不会浪费在只是为这些事件做准备的过度供应上。应用程序负载预测引擎(例如结合图1描述的应用程序负载预测引擎114)可以预测应用程序负载和季节性。

[0062] 流程图500继续到学习应用程序在负载下的行为功能的模块506。动态估计引擎(例如结合图1描述的动态估计引擎112)可以学习应用程序在负载下的行为功能。

[0063] 流程图500继续到估计在资源请求的预测需求下使用的资源的模块508。应用程序负载预测引擎(例如结合图1描述的应用程序负载预测引擎114)可以估计在资源请求的预测需求下使用的资源。

[0064] 流程图500继续到估计用于设置资源限制的预测模式的模块510。应用程序负载预测引擎(例如结合图1描述的应用程序负载预测引擎114)可以估计用于设置资源限制的预测模式。

[0065] 流程图500继续到使满足预测需求所需的资源最小化的模块512。最低成本优化引

擎(例如结合图1描述的最低成本优化引擎116)可以最小化满足预测需求所需的资源。

[0066] 流程图500在根据做出的决策学习以改进预测和资源估计的模块514处结束。扩展资源最佳配置执行器引擎(例如结合图1描述的扩展资源最佳配置执行器引擎118)可以在收敛部署中从根据做出的决策学习以改进预测和资源估计中受益。

[0067] 图6描绘了用于生成最低成本优化参数的系统的例子的图表600。图表600包括:SLA指标数据存储区604,其可以被实现为结合图1描述的SLA指标数据存储区104;声明式性能接口引擎608,其与SLA指标数据存储区604耦合并且可以被实现为结合图1描述的声明式性能接口引擎108;动力学估计引擎612,其可以被实现为结合图1描述的动力学估计引擎112;应用程序负载预测引擎614,其可以被实现为结合图1描述的应用程序负载预测引擎114;最低成本优化引擎616,其可以被实现为结合图1描述的最低成本优化引擎116;声明式性能数据存储区626,其与声明式性能接口引擎608和最低成本优化引擎616耦合;行为模型数据存储区628,其与动态估计引擎612和最低成本优化引擎616耦合;性能模型数据存储区630,其与动态估计引擎612和最低成本优化引擎616耦合;收敛部署和资源指标数据存储区632,其与动态估计引擎612和应用程序负载预测引擎614耦合;利用模式学习引擎634,其与收敛部署和资源指标数据存储区632耦合;预测模型数据存储区636,其与应用程序负载预测引擎614和利用模式学习引擎634耦合;预测负载数据存储区638,其与应用程序负载预测引擎614和最低成本优化引擎616耦合;以及最低成本优化参数数据存储区640,其与最低成本优化引擎616耦合。

[0068] 声明式性能接口引擎608将来自SLA指标数据存储区604的SLA指标转换为由声明式性能数据存储区626表示的声明式性能数据结构。声明式性能接口引擎608可以或可以不接收来自服务提供者(或服务消费者)的人类或人工代理的指令以填充SLA指标数据存储区604。如果SLA指标数据存储区604被修改,声明式性能引擎608转换修改以匹配在SLA指标数据存储区604中用声明式性能数据存储区626中的声明式性能数据结构表示的预期的SLO。

[0069] 动态估计引擎608使用诸如深度学习的机器学习技术来生成由行为模型数据存储区628表示的行为模型并生成由性能模型数据存储区630表示的性能模型。可以通过与适用的收敛部署相关联的反馈来改进模型。这种反馈由收敛部署和资源指标数据存储区632表示。收敛部署和资源指标数据存储区632可以由收敛部署和资源指标收集引擎(未被示出)填充,该收敛部署和资源指标收集引擎可以被实现为结合图1描述的收敛部署、资源和应用程序级别指标收集引擎120。

[0070] 利用模式学习引擎634使用深度学习来理解工作负载以生成用于季节性负载、流行负载、突发负载和随机负载的模型。基于请求和对工作负载的理解,可以根据利用模式在部署上设置签名限制(例如,突发导致较高的限制,而稳定导致较低的限制)。深度学习的结果是由预测模型数据存储区636表示的预测模型。可以利用与适用的收敛部署相关联的反馈来改进预测模型数据存储区636。这种反馈由收敛部署和资源指标数据存储区632表示。

[0071] 应用程序负载预测引擎614使用来自预测模型数据存储区636的一个或多个预测模型以及来自收敛部署和资源指标数据存储区632的反馈来估计未来时间的资源使用;该预测负载由预测的负载数据存储区638表示。

[0072] 最低成本优化引擎616使用声明式性能数据存储区626、行为模型数据存储区628、性能模型数据存储区630和预测负载数据存储区638来生成由最低成本优化参数数据存储

区640表示的最低成本优化参数。最低成本优化参数可以由软件部署平台使用，该软件部署平台可以包括例如扩展资源最佳配置执行器引擎(未被示出)，其可以被实现为结合图1描述的扩展资源最佳配置执行器引擎118。

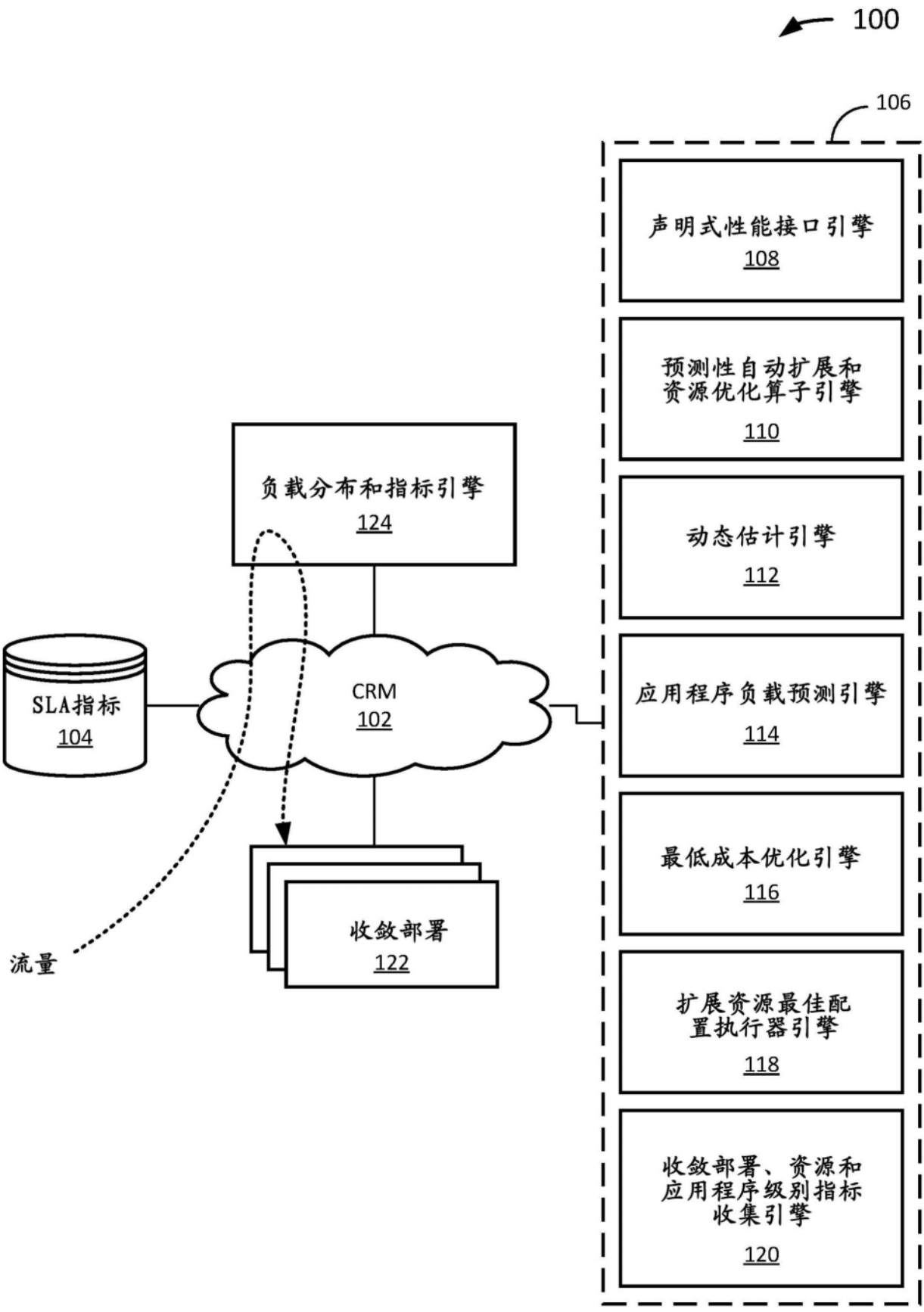

100
106
声明式性能接口引擎
108
预测性自动扩展和
资源优化算子引擎
110
动态估计引擎
112
应用程序负载预测引擎
114
最低成本优化引擎
116
扩展资源最佳配
置执行器引擎
118
收敛部署、资源和
应用程序级别指标
收集引擎
120
负载分布和指标引擎
124
SLA指标
104
CRM
102
收敛部署
122
流量

图1

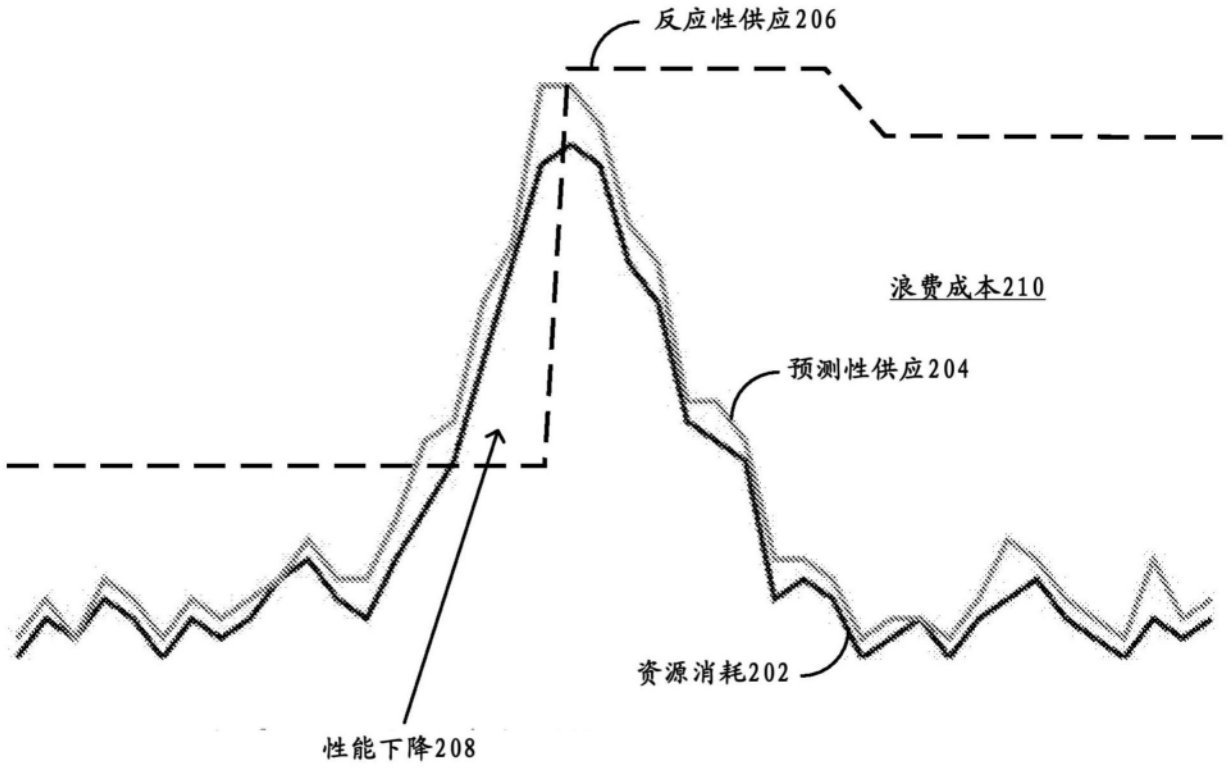

反应性供应206
浪费成本210
预测性供应204
资源消耗202
性能下降208

图2

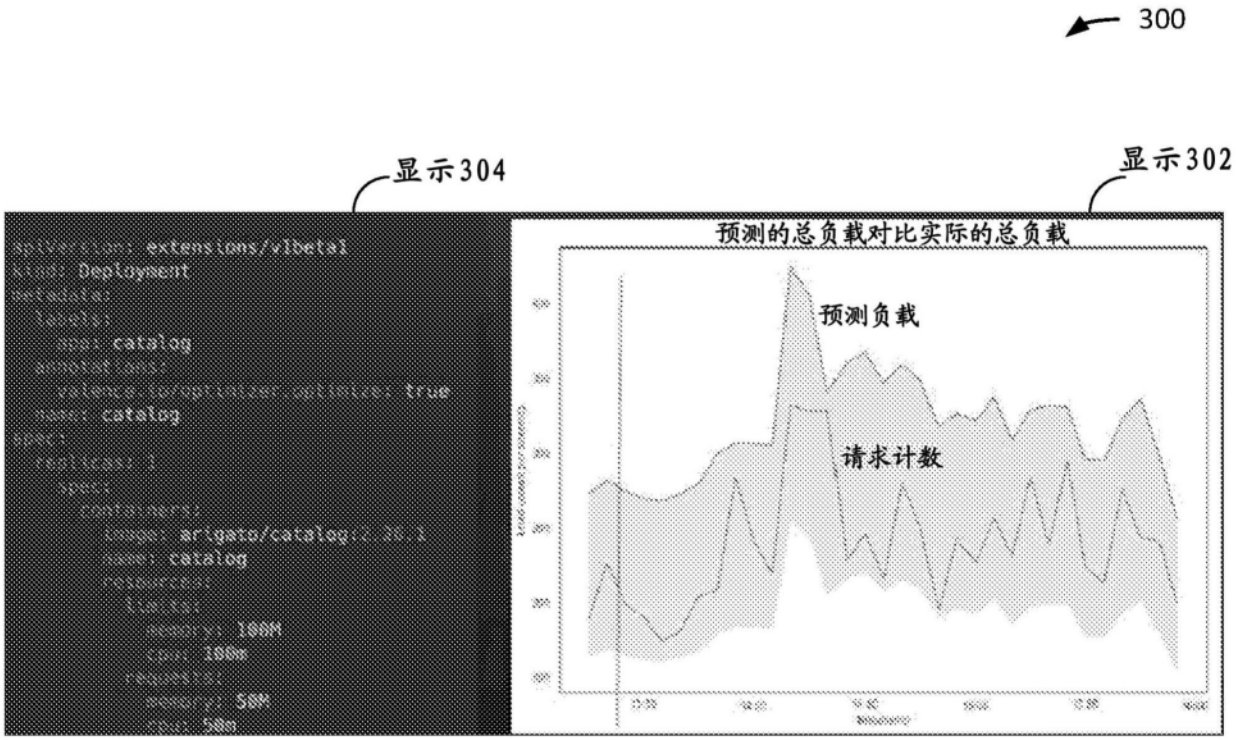

300
显示304
显示302
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
labels:
app: catalog
annotations:
name: catalog
spec:
replicas: 1
template:
containers:
image: arigato/catalog:2.35.1
name: catalog
resources:
limits:
memory: 100M
cpu: 100m
requests:
memory: 50M
cpu: 50m
预测的总负载对比实际的总负载
预测负载
请求计数

图3

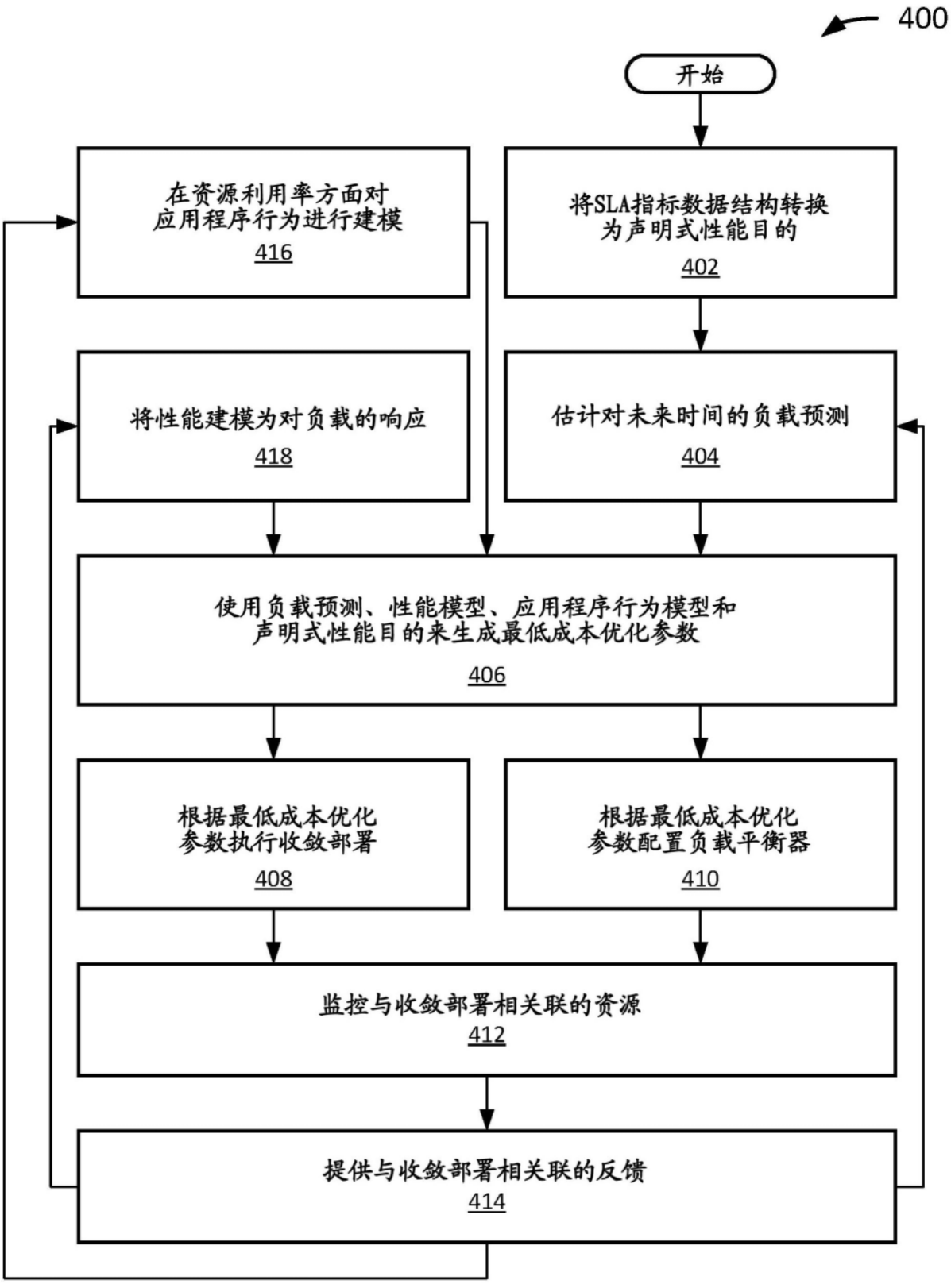

400
开始
在资源利用率方面对
应用程序行为进行建模
416
将SLA指标数据结构转换
为声明式性能目的
402
将性能建模为对负载的响应
418
估计对未来时间的负载预测
404
使用负载预测、性能模型、应用程序行为模型和
声明式性能目的来生成最低成本优化参数
406
根据最低成本优化
参数执行收敛部署
408
根据最低成本优化
参数配置负载平衡器
410
监控与收敛部署相关联的资源
412
提供与收敛部署相关联的反馈
414

图4

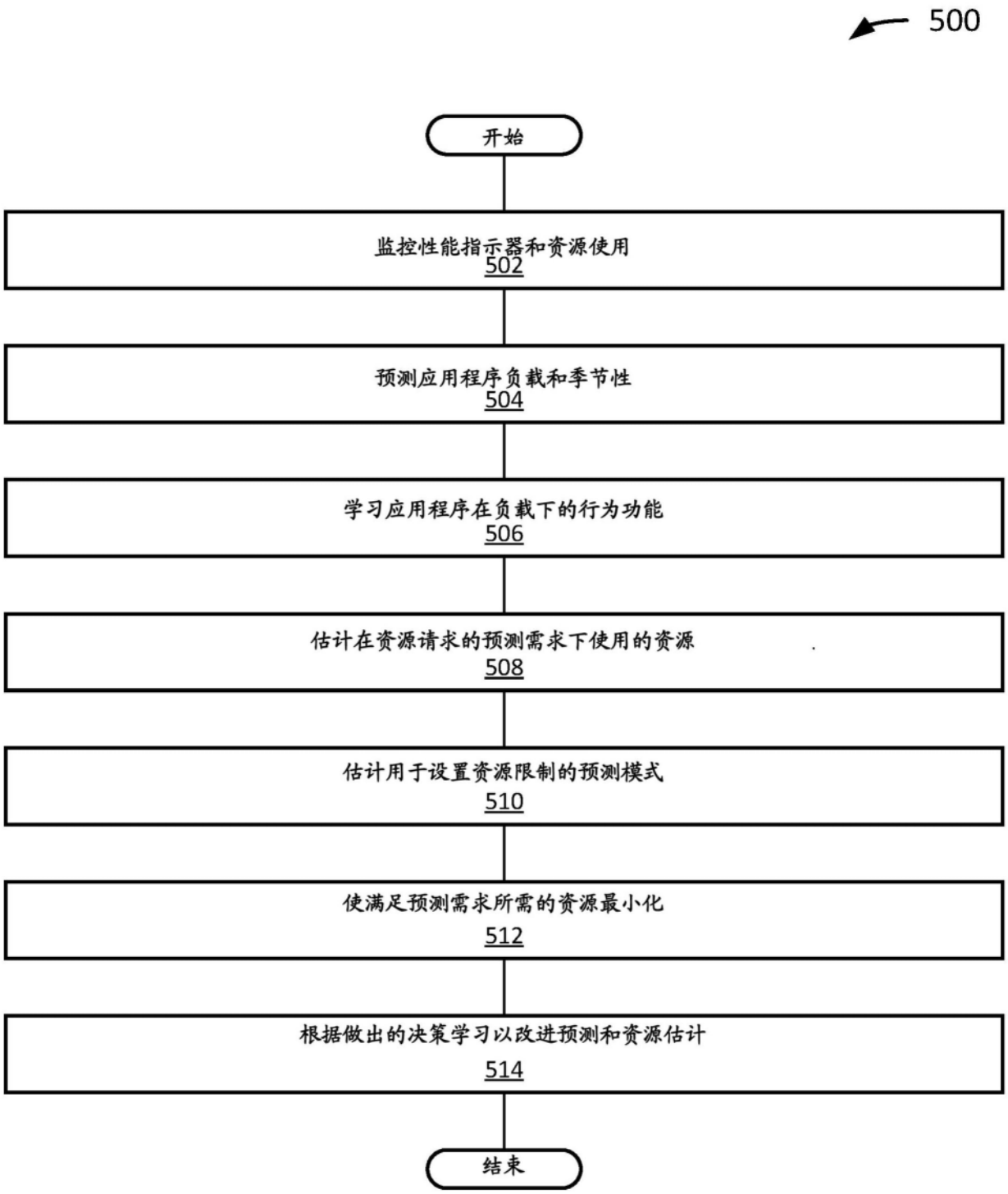

500
开始
监控性能指示器和资源使用
502
预测应用程序负载和季节性
504
学习应用程序在负载下的行为功能
506
估计在资源请求的预测需求下使用的资源
508
估计用于设置资源限制的预测模式
510
使满足预测需求所需的资源最小化
512
根据做出的决策学习以改进预测和资源估计
514
结束

图5

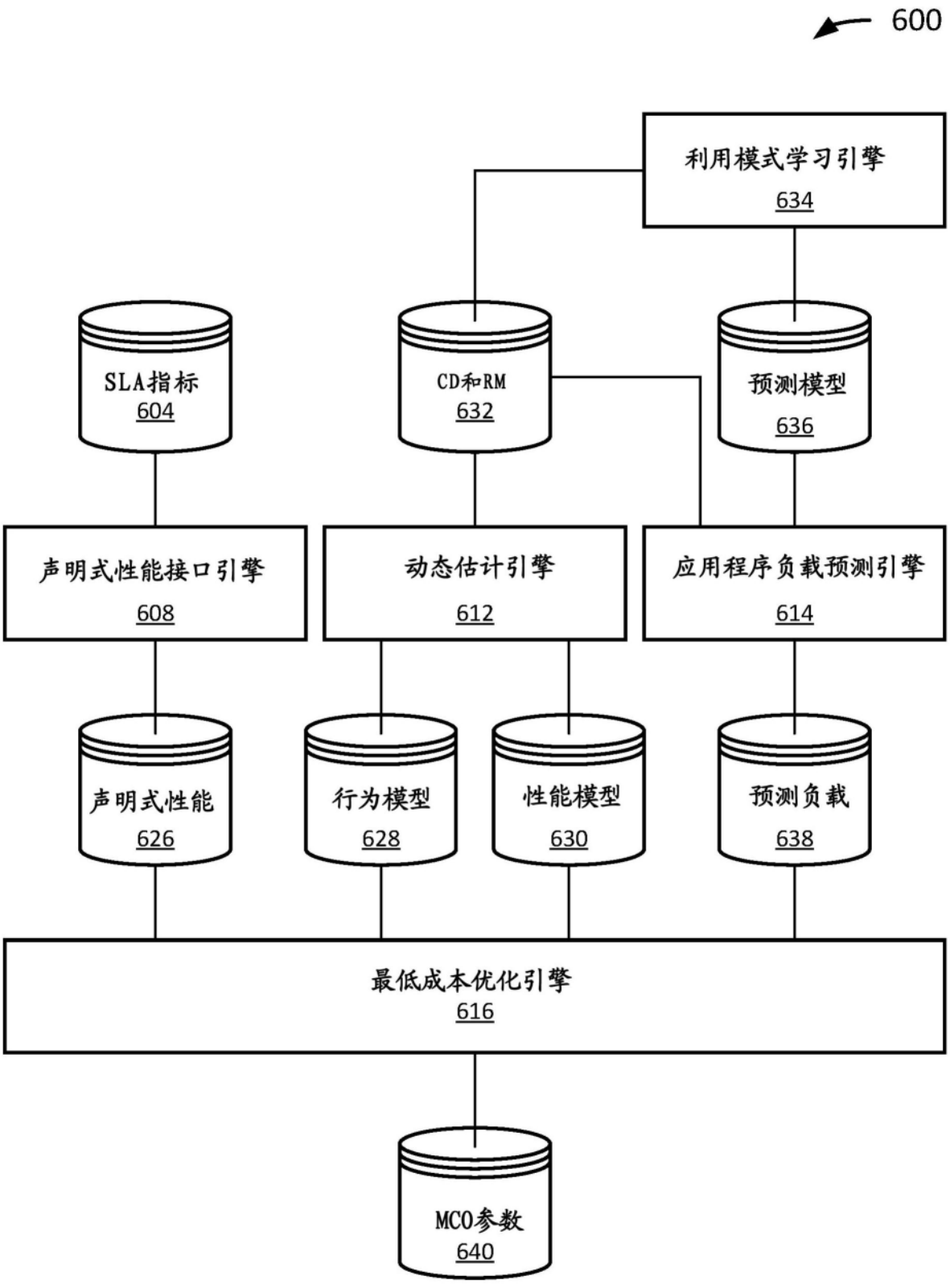

600
利用模式学习引擎
634
SLA指标
604
CD和RM
632
预测模型
636
声明式性能接口引擎
608
动态估计引擎
612
应用程序负载预测引擎
614
声明式性能
626
行为模型
628
性能模型
630
预测负载
638
最低成本优化引擎
616
MCO参数
640

图6