(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 122412063 A
(43)申请公布日 2026.07.17

(21)申请号 202511958172.4

(22)申请日 2025.12.23

(71)申请人 中煤科工开采研究院有限公司
地址 101399 北京市顺义区中关村科技园区顺义园临空二路1号

(72)发明人 吕依濛

(74)专利代理机构 北京清亦华知识产权代理事务所(普通合伙) 11201
专利代理师 宋圆圆

(51)Int.Cl.
G06F 9/455(2018.01)
G06F 9/50(2006.01)

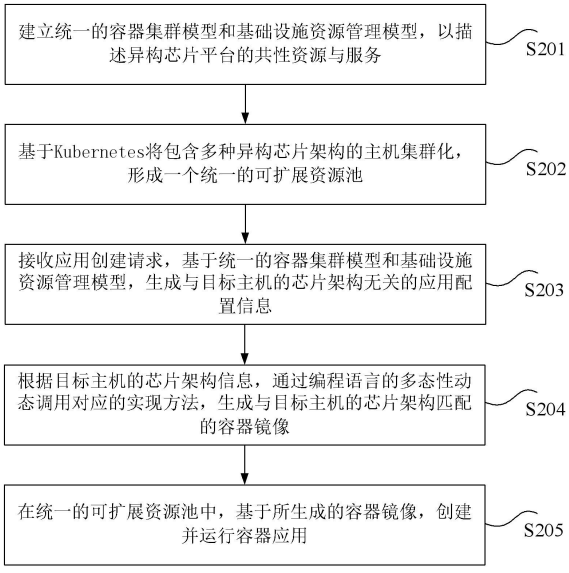
权利要求书2页 说明书13页 附图3页

(54)发明名称

一种异构芯片容器云平台的统一管理方法及装置

(57)摘要

本申请涉及云计算及数据中心基础设施管理技术领域,具体涉及一种异构芯片容器云平台的统一管理方法及装置,本申请通过构建架构无关的统一抽象层(模型与资源池),并在运行时动态绑定具体架构实现(多态性生成镜像),从根本上解决了ARM、MIPS、LoongArch、x86等异构芯片无法在同一容器云平台下统一管理、调度和运维的业界难题,实现了资源池化、管理标准化和运维简化,显著提高了资源利用率和降低了采购与运维成本。



CN 122412063 A

1.一种异构芯片容器云平台的统一管理方法,其特征在于,包括:

建立统一的容器集群模型和基础设施资源管理模型,以描述异构芯片平台的共性资源与服务;

基于Kubernetes将包含多种异构芯片架构的主机集群化,形成一个统一的可扩展资源池;

接收应用创建请求,基于所述统一的容器集群模型和基础设施资源管理模型,生成与目标主机的芯片架构无关的应用配置信息;

根据所述目标主机的芯片架构信息,通过编程语言的多态性动态调用对应的实现方法,生成与所述目标主机的芯片架构匹配的容器镜像;

在所述统一的可扩展资源池中,基于所生成的容器镜像,创建并运行容器应用。

2.根据权利要求1所述的异构芯片容器云平台的统一管理方法,其特征在于,所述基础设施资源管理模型通过综合参数p来衡量一个应用所需的资源量,并根据预设的极值x和y,将资源需求划分为n个级别l,其中级别l的计算公式为:

$$l=\left\lfloor \frac{y-p}{y-x}n \right\rfloor$$

式中,l为系统资源级别,为1到n之间的整。

3.根据权利要求2所述的异构芯片容器云平台的统一管理方法,其特征在于,所述综合参数p通过以下公式确定:

$$p=\frac{\sum_{t_3=0}\frac{(-1)^{t_3-1}}{t_3}\sum_{t_2=0}\frac{1}{t_3 2^{t_2}+1}\int_0^{t_1^2}\frac{\pi(\sqrt[4]{1+a}-1)\sin a^4}{\sum_{t_3}^{\infty}\frac{((t_3-1)!)^2(2t_1)^{2t_3}}{(2t_3)!}\int_0^1\frac{(1-2t_1)In(1-t_1)}{t_1^2-t_1+1}dt_1}dt_1}{t_1^2(t_1-\tan t_1)In(t_1^2-1)[(\frac{2\arctan\frac{t_4}{t_1}}{\pi})^{t_4}-1]}$$

式中,t_1为计算资源,t_2为网络资源,t_3为存储资源,t_4为所需docker附属资源,a为根据系统实际使用设定的参数,用来协调系统对于资源规模的认定,其中t1,t2,t3,t4为正整数,-1<=a<=1。

4.根据权利要求3所述的异构芯片容器云平台的统一管理方法,其特征在于,极值x和y分别通过以下公式确定:

$$x=\lim_{x\to+\infty}\lim_{y\to0}p$$

$$y=\lim_{x\to0}\lim_{y\to+\infty}p$$

式中,P为综合参数。

5.根据权利要求1所述的异构芯片容器云平台的统一管理方法,其特征在于,所述方法还包括:

建立统一的主机资源管理模型;

基于所述主机资源管理模型评估主机的负载状态,从所述统一的可扩展资源池中筛选出负载值r小于预设阈值的主机,构成待分配主机集合,其中,负载值r通过以下公式确定:

$$r = \frac{s_2}{s_1}a_1 + \frac{s_3}{s_1}a_2 + \frac{s_2 + s_3}{s_1}a_3$$

式中，s_1为主机资源，s_2为已用资源，s_3为资源所用存储；a_1，a_2，a_3为三个0~1之间的调节参数分别视主机剩余资源，待分配目标资源，整体占用资源的重要程度设定。

6.根据权利要求5所述的异构芯片容器云平台的统一管理方法，其特征在于，在生成与所述目标主机的芯片架构匹配的容器镜像前，所述方法包括：

基于一个与芯片性能相关的函数F(x)来计算一个优先级参数λ，其中上述计算步骤用以下公式表示：

$$F(x) = 4n(\frac{\pi}{4} - \sum_{1}^{x}\frac{n}{n^2 + x^2})$$

$$\lambda = b_1 F(x) + b_2$$

式中x的取值是n，b1，b2用来调节弹性伸缩等对于资源的影响；

将所述优先级参数λ与不同的芯片架构映射；

根据所述优先级参数λ值对所述待分配主机集合中的主机进行优先级排序，并选择优先级最高的主机用于创建容器应用。

7.根据权利要求1所述的异构芯片容器云平台的统一管理方法，其特征在于，所述统一的容器集群模型在进行扩展时，遵循只增加属性或方法的原则，以保证对系统的向后兼容性；

在运行容器应用过程中，使用Docker提供底层容器虚拟化，与所述目标主机的芯片架构匹配的容器镜像由Docker拉取和运行。

8.一种异构芯片容器云平台的统一管理装置，其特征在于，包括：

模型管理模块，被配置为建立统一的容器集群模型和基础设施资源管理模型，以描述异构芯片平台的共性资源与服务；

集群化模块，被配置为基于Kubernetes将包含多种异构芯片架构的主机集群化，形成一个统一的可扩展资源池；

应用调度模块，被配置为接收应用创建请求，基于所述统一的容器集群模型和基础设施资源管理模型，生成与目标主机的芯片架构无关的应用配置信息；

镜像构建模块，被配置为根据所述目标主机的芯片架构信息，通过编程语言的多态性动态调用对应的实现方法，生成与所述目标主机的芯片架构匹配的容器镜像；

容器运行时模块，被配置为在所述统一的可扩展资源池中，基于所生成的容器镜像，创建并运行容器应用。

9.一种电子设备，包括处理器、通信接口、存储器和通信总线，其中，所述处理器、所述通信接口和所述存储器通过所述通信总线完成相互间的通信，其特征在于，

所述存储器，用于存储计算机程序；

所述处理器，用于通过运行所述存储器上所存储的所述计算机程序来执行权利要求1至7中任一项所述的方法步骤。

10.一种计算机可读的存储介质，其特征在于，所述存储介质中存储有计算机程序，其中，所述计算机程序被设置为运行时执行权利要求1至7中任一项中所述的方法步骤。

一种异构芯片容器云平台的统一管理方法及装置

技术领域

[0001] 本申请涉及云计算及数据中心基础设施管理技术领域,具体涉及一种异构芯片容器云平台的统一管理方法及装置。

背景技术

[0002] 容器云平台作为融合IaaS和PaaS能力的云原生核心设施,通过容器、编排、服务网格等技术,为应用提供了全生命周期的管理,已成为支撑企业数字化转型的关键基础设施。随着技术发展,各类业务架构正加速向容器化和云原生迁移。

[0003] 然而,现有的容器云平台通常仅针对单一芯片架构(如X86)进行设计和优化。而随着信息技术应用创新的推进,用户数据中心普遍存在多种国产化芯片(如ARM、MIPS、LoongArch)与X86芯片共存的异构环境。由于这些芯片的指令集、存储和网络访问存在固有差异,导致基于不同架构的容器镜像无法兼容运行,使得现有平台难以对异构计算资源进行统一管控。

[0004] 这一根本缺陷迫使用户为不同架构的服务器集群分别采购和部署独立的容器云平台,形成了多个资源孤岛。这不仅造成了采购和建设成本的显著增加,更导致了计算资源的浪费与管理运维的复杂化,无法实现资源的集中调度与高效利用,严重制约了企业基础设施的灵活性和整体效能。。

发明内容

[0005] 本申请的目的在于提供一种异构芯片容器云平台的统一管理方法及装置,以解决现有技术中当前容器云平台无法跨不同芯片架构实现统一管理的问题。

[0006] 为了实现上述目的,本申请采用的技术方案如下:

根据本申请实施例的一个方面,提供了一种异构芯片容器云平台的统一管理方法,包括:建立统一的容器集群模型和基础设施资源管理模型,以描述异构芯片平台的共性资源与服务;基于Kubernetes将包含多种异构芯片架构的主机集群化,形成一个统一的可扩展资源池;接收应用创建请求,基于统一的容器集群模型和基础设施资源管理模型,生成与目标主机的芯片架构无关的应用配置信息;根据目标主机的芯片架构信息,通过编程语言的多态性动态调用对应的实现方法,生成与目标主机的芯片架构匹配的容器镜像;在统一的可扩展资源池中,基于所生成的容器镜像,创建并运行容器应用。

[0007] 根据上述技术手段,通过构建架构无关的统一抽象层(模型与资源池),并在运行时动态绑定具体架构实现(多态性生成镜像),从根本上解决了ARM、MIPS、LoongArch、x86等异构芯片无法在同一容器云平台下统一管理、调度和运维的业界难题,实现了资源池化、管理标准化和运维简化,显著提高了资源利用率和降低了采购与运维成本。

[0008] 进一步,基础设施资源管理模型通过综合参数p来衡量一个应用所需的资源量,并根据预设的极值x和y,将资源需求划分为n个级别l,其中级别l的计算公式为:

$$l=\left\lfloor \frac{y-p}{y-x} n \right\rfloor$$

[0009] 式中,l为系统资源级别,为1到n之间的整。

[0010] 根据上述技术手段,将原本异构、定性的资源需求转化为一个可计算的量化参数p,并进一步分级,使得跨架构的资源调度有了一套统一的、可度量的标准,为后续的自动化、精细化调度奠定了数学基础,提升了资源分配的公平性和效率。

[0011] 进一步,综合参数p通过以下公式确定:

$$p=\frac{\sum_{t_3=0}\frac{(-1)^{t_3-1}}{t_3}\sum_{t_2=0}\frac{1}{t_3 2^{t_2}+1}\int_0^{t_1^2}\frac{\pi(\sqrt[4]{1+a}-1)\sin a^4}{\sum_{t_3}^{\infty}\frac{((t_3-1)!)^2(2t_1)^{2t_3}}{(2t_3)!}\int_0^1\frac{(1-2t_1)In(1-t_1)}{t_1^2-t_1+1}dt_1}dt_1}{t_1^2(t_1-\tan t_1)In(t_1^2-1)[(\frac{2\arctan\frac{t_4}{t_1}}{\pi})^{t_4}-1]}$$

[0012] 式中,t_1为计算资源,t_2为网络资源,t_3为存储资源,t_4为所需docker附属资源,a为根据系统实际使用设定的参数,用来协调系统对于资源规模的认定,其中t1,t2,t3,t4为正整数,-1<=a<=1。

[0013] 进一步,极值x和y分别通过以下公式确定:

$$x=\lim_{x\to+\infty}\lim_{y\to 0} p$$

[0014] $$y=\lim_{x\to 0}\lim_{y\to+\infty} p$$

[0015] 式中,P为综合参数。

[0016] 根据上述技术手段,为上述量化参数的量化与分级模型提供了完整、具体的数学定义。通过明确的公式(尽管复杂)固定了资源衡量的算法,使得“统一管理”不再是概念上的,而是具有确定性和可重复性的技术方案,增强了发明的可实现性和保护范围的具体性,防止了因描述模糊而被轻易绕过。

[0017] 进一步,上述方法还包括:建立统一的主机资源管理模型;基于主机资源管理模型评估主机的负载状态,从统一的可扩展资源池中筛选出负载值r小于预设阈值的主机,构成待分配主机集合,其中,负载值r通过以下公式确定:

$$r=\frac{s_2}{s_1}a_1+\frac{s_3}{s_1}a_2+\frac{s_2+s_3}{s_1}a_3$$

[0018] 式中,s_1为主机资源,s_2为已用资源,s_3为资源所用存储;a_1,a_2,a_3为三个0~1之间的调节参数分别视主机剩余资源,待分配目标资源,整体占用资源的重要程度设定。

[0019] 根据上述技术手段,引入了针对主机粒度的统一管理模型和量化筛选机制。通过综合考虑主机剩余资源、目标资源需求和整体负载,公式r能够更智能、更全面地评估主机状态,避免将新应用调度到已过载或存储紧张的主机上,从而提升了集群整体的稳定性和性能。

[0020] 进一步,在生成与所述目标主机的芯片架构匹配的容器镜像前,该方法包括:基于一个与芯片性能相关的函数F(x)来计算一个优先级参数λ,其中上述计算步骤用以下公式表示:

$$F(x) = 4n(\frac{\pi}{4} - \sum_{1}^{x} \frac{n}{n^2 + x^2})$$

[0021] $$\lambda = b_1 F(x) + b_2$$

[0022] 式中x的取值是n,b1,b2用来调节弹性伸缩等对于资源的影响;将优先级参数λ与不同的芯片架构映射;根据优先级参数λ值对待分配主机集合中的主机进行优先级排序,并选择优先级最高的主机用于创建容器应用。

[0023] 根据上述技术手段,在主机筛选的基础上,增加了基于芯片架构性能的智能选择策略。通过函数F(x)和参数λ,系统能够识别不同架构芯片的性能特性差异,并结合弹性伸缩等需求,优先将任务调度到最合适的芯片架构上,实现了从“能用”到“优用”的跨越,进一步优化了应用性能和资源效能。

[0024] 进一步,统一的容器集群模型在进行扩展时,遵循只增加属性或方法的原则,以保证对系统的向后兼容性;在运行容器应用过程中,使用Docker提供底层容器虚拟化,与目标主机的芯片架构匹配的容器镜像由Docker拉取和运行。

[0025] 根据上述技术手段,明确底层依赖(Docker)确保了方案的通用性和可实施性;规定模型的扩展原则(只增不改)则保证了该统一管理平台具有良好的演进性和生态兼容性。当未来出现新的芯片架构时,可以平滑接入而不影响现有系统,极大地延长了平台的生命周期。

[0026] 根据本申请实施例的又一个方面,还提供了一种异构芯片容器云平台的统一管理装置,包括:模型管理模块,被配置为建立统一的容器集群模型和基础设施资源管理模型,以描述异构芯片平台的共性资源与服务;集群化模块,被配置为基于Kubernetes将包含多种异构芯片架构的主机集群化,形成一个统一的可扩展资源池;应用调度模块,被配置为接收应用创建请求,基于统一的容器集群模型和基础设施资源管理模型,生成与目标主机的芯片架构无关的应用配置信息;镜像构建模块,被配置为根据目标主机的芯片架构信息,通过编程语言的多态性动态调用对应的实现方法,生成与目标主机的芯片架构匹配的容器镜像;容器运行时模块,被配置为在统一的可扩展资源池中,基于所生成的容器镜像,创建并运行容器应用。

[0027] 根据本申请实施例的又一个方面,还提供了一种电子设备,包括处理器、通信接口、存储器和通信总线,其中,处理器、通信接口和存储器通过通信总线完成相互间的通信;其中,存储器,用于存储计算机程序;处理器,用于通过运行存储器上所存储的计算机程序来执行上述任一实施例中的异构芯片容器云平台的统一管理方法方法步骤。

[0028] 根据本申请实施例的又一个方面,还提供了一种计算机可读的存储介质,该存储介质中存储有计算机程序,其中,该计算机程序被设置为运行时执行上述任一实施例中的异构芯片容器云平台的统一管理方法方法步骤。

[0029] 本申请的有益效果:

本申请通过构建架构无关的统一抽象层(模型与资源池),并在运行时动态绑定具体架构实现(多态性生成镜像),从根本上解决了ARM、MIPS、LoongArch、x86等异构芯片无法在同一容器云平台下统一管理、调度和运维的业界难题,实现了资源池化、管理标准化和运维简化,显著提高了资源利用率和降低了采购与运维成本。

附图说明

[0030] 此处的附图被并入说明书中并构成本说明书的一部分,示出了符合本申请的实施例,并与说明书一起用于解释本申请的原理。

[0031] 为了更清楚地说明本申请实施例或现有技术中的技术方案,下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍,显而易见地,对于本领域普通技术人员而言,在不付出创造性劳动性的前提下,还可以根据这些附图获得其他的附图。

[0032] 图1是本申请实施例提供的一种可选的异构芯片容器云平台的统一管理方法的硬件环境的示意图;

图2是本申请实施例提供的一种可选的异构芯片容器云平台的统一管理方法的流程示意图;

图3是本申请实施例提供的一种可选的异构芯片容器云平台的统一管理装置的结构框图;

图4是本申请实施例提供的一种可选的电子设备的结构框图;

图5是本申请实施例提供的另一种可选的异构芯片容器云平台的统一管理方法的流程示意图。

具体实施方式

[0033] 为了使本技术领域的人员更好地理解本申请方案,下面将结合本申请实施例中的附图,对本申请实施例中的技术方案进行清楚、完整地描述,显然,所描述的实施例仅仅是本申请一部分的实施例,而不是全部的实施例。基于本申请中的实施例,本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例,都应当属于本申请保护的范围。

[0034] 需要说明的是,本申请的说明书和权利要求书及上述附图中的术语“第一”、“第二”等是用于区别类似的对象,而不必用于描述特定的顺序或先后次序。应该理解这样使用的数据在适当情况下可以互换,以便这里描述的本申请的实施例能够以除了在这里图示或描述的那些以外的顺序实施。此外,术语“包括”和“具有”以及他们的任何变形,意图在于覆盖不排他的包含,例如,包含了一系列步骤或单元的过程、方法、系统、产品或设备不必限于清楚地列出的那些步骤或单元,而是可包括没有清楚地列出的或对于这些过程、方法、产品或设备固有的其它步骤或单元。

[0035] 根据本申请实施例的一个方面,提供了一种异构芯片容器云平台的统一管理方法。可选地,在本实施例中,上述异构芯片容器云平台的统一管理方法可以应用于由终端和服务器所构成的硬件环境中。服务器通过网络与终端进行连接,可用于为终端或终端上安装的客户端提供服务,可在服务器上或独立于服务器设置数据库,用于为服务器提供数据存储服务。

[0036] 上述网络可以包括但不限于以下至少之一:有线网络,无线网络。上述有线网络可以包括但不限于以下至少之一:广域网,城域网,局域网,上述无线网络可以包括但不限于以下至少之一:WIFI(Wireless Fidelity,无线保真),蓝牙。终端可以并不限定于为PC、手机、平板电脑等。

[0037] 本申请实施例的异构芯片容器云平台的统一管理方法可以由服务器来执行,也可

以由终端来执行,还可以是由服务器和终端共同执行。其中,终端执行本申请实施例的异构芯片容器云平台的统一管理方法也可以是由安装在其上的客户端来执行。

[0038] 以由服务器来执行本实施例中的异构芯片容器云平台的统一管理方法为例,请参阅图1,图1是本申请实施例提供的一种可选的异构芯片容器云平台的统一管理方法的硬件环境的示意图,如图1所示,该异构芯片容器云平台的统一管理方法的硬件环境包括:终端102,以及通过网络与终端102连接的服务器104。服务器104用于统一的容器云平台管理模型与核心调度引擎,该平台执行本申请实施例的异构芯片容器云平台的统一管理方法,对异构计算资源进行统一纳管与容器化应用调度;终端102用于展示平台管理界面、资源状态监控信息与应用部署结果,其中,该信息与结果可以由服务器104部署的平台进行处理和提供。

[0039] 本实施例中的异构芯片容器云平台的统一管理方法可以应用于包含多种国产化芯片(如ARM、MIPS、LoongArch、SW)与X86芯片混合数据中心的统一运维、企业在信创转型过程中对新旧异构IT资源的整合管理、云服务商提供跨架构兼容的容器云服务等场景,例如:金融、政务、能源等关键行业建设自主可控云平台等。本申请实施例中以在混合架构数据中心中部署和管理一套容器化应用为例说明上述的异构芯片容器云平台的统一管理方法。

[0040] 现有技术中存在的问题在于,主流的容器云平台通常仅针对单一的芯片架构(如X86)进行设计与优化,无法有效纳管指令集与系统特性各异的多种国产化芯片(如ARM、MIPS、LoongArch等)与X86芯片共存的环境。这导致用户在面对异构计算资源时,不得不为不同架构的服务器集群分别部署和管理独立的容器云平台,形成了资源与管理上的“孤岛”,从而造成计算资源利用率低下、采购与运维成本倍增、无法实现应用的统一调度与全局监控等一系列问题。

[0041] 为解决上述问题,本实施例中提供了一种运行于上述服务器的异构芯片容器云平台的统一管理方法,请参阅图2,图2是本申请实施例提供的一种可选的异构芯片容器云平台的统一管理方法的流程示意图,如图2所示,本申请实施例的异构芯片容器云平台的统一管理方法具体包括如下步骤:

步骤S201,建立统一的容器集群模型和基础设施资源管理模型,以描述异构芯片平台的共性资源与服务;

步骤S202,基于Kubernetes将包含多种异构芯片架构的主机集群化,形成一个统一的可扩展资源池;

步骤S203,接收应用创建请求,基于统一的容器集群模型和基础设施资源管理模型,生成与目标主机的芯片架构无关的应用配置信息;

步骤S204,根据目标主机的芯片架构信息,通过编程语言的多态性动态调用对应的实现方法,生成与目标主机的芯片架构匹配的容器镜像;

步骤S205,在统一的可扩展资源池中,基于所生成的容器镜像,创建并运行容器应用。

[0042] 通过上述步骤S201至步骤S205,通过构建架构无关的统一抽象层(模型与资源池),并在运行时动态绑定具体架构实现(多态性生成镜像),从根本上解决了ARM、MIPS、LoongArch、x86等异构芯片无法在同一容器云平台下统一管理、调度和运维的业界难题,实现了资源池化、管理标准化和运维简化,显著提高了资源利用率和降低了采购与运维成本。

[0043] 如图5所示,该流程图直观地阐释了本方案异构芯片容器云平台统一管理方法的核心构建逻辑与流程。其过程始于对管理模型的统一抽象:首先定义统一的多集群管理模型,以实现对分散资源的逻辑集中;随后定义统一的主机资源管理模型,以标准化方式刻画各主机的资源状态。关键步骤在于利用多态编程定义统一的实现接口,这为屏蔽底层差异奠定了技术基础。最后,流程通过一个循环扩展机制,展示了该统一模型可具体适配并管理的一系列异构芯片架构,包括SW、C86(海光)、ARM(华为鲲鹏/飞腾)、MIPS、X86等,从而形象地说明了本发明如何通过“统一抽象接口”与“多态具体实现”相结合,将多样的异构芯片资源纳入一套管理体系之中。

[0044] 下面结合图2对本申请实施例中的异构芯片容器云平台的统一管理方法进行解释说明。

[0045] 异构芯片指指令集架构(ISA)不同的中央处理器(CPU)。在本实施例中,特指需要统一管理的多种国产化CPU(如ARM、MIPS、LoongArch、SW、C86)与国际主流的X86架构CPU共存的计算环境。例如:一个数据中心可能同时部署了基于ARM架构的华为鲲鹏服务器、基于LoongArch架构的龙芯服务器以及传统的Intel X86服务器。这些服务器中的CPU即为“异构芯片”。

[0046] 容器云平台是一种基于容器技术、云原生理念构建的PaaS(平台即服务)层软件系统。它融合了基础设施(IaaS)和平台能力,提供从应用开发、编排、部署到运维监控的全生命周期管理服务,其核心支撑技术包括容器运行时(如Docker)和容器编排系统(如Kubernetes)。

[0047] 在步骤S201的技术方案中,建立统一的容器集群模型和基础设施资源管理模型,以描述异构芯片平台的共性资源与服务。

[0048] 统一的容器集群模型是一种用于抽象描述容器集群逻辑状态和属性的数据模型。它定义了一套标准的对象和关系,用于表示集群中的节点、应用、服务等实体,旨在屏蔽底层异构芯片的差异,为上层管理提供一致的视图和操作接口。例如:该模型会定义“应用(Application)”、“服务(Service)”、“实例(Pod)”等标准对象,无论底层是ARM还是X86服务器,一个“应用”对象都包含相同的核心属性字段(如名称、副本数、资源请求),但其具体部署时生成的镜像和二进制文件会因架构而异。

[0049] 容器集群统一管理模型包括标准对象和扩展对象。其中,标准对象定义了容器集群的共性特征,涵盖了Kubernetes原生定义的对象;扩展对象则用于描述标准对象中未统一定义的、各异构平台特有的属性,从而保证了模型的通用性与可扩展性。

[0050] 基础设施资源管理模型是一种用于量化、描述和调度底层物理(或虚拟)资源的抽象模型。它将异构的计算、存储、网络资源转化为统一的、可度量、可比较的量化指标和分级标准,是实现跨架构资源统一调度和分配的数学基础。例如:该模型通过一个综合公式计算参数p来代表一个应用的整体资源需求强度,并将p值映射到预定义的资源级别l(如分为“小”、“中”、“大”、“特大”四级),从而忽略具体架构差异,实现按“资源级别”进行调度。

[0051] 作为一种可选的实施例,基础设施资源管理模型通过综合参数p来衡量一个应用所需的资源量,并根据预设的极值x和y,将资源需求划分为n个级别l,其中级别l的计算公式为:

$$l=\left\lfloor \frac{y-p}{y-x}n \right\rfloor$$

[0052] 式中,l为系统资源级别,为1到n之间的整。

[0053] 根据上述技术手段,将原本异构、定性的资源需求转化为一个可计算的量化参数p,并进一步分级,使得跨架构的资源调度有了一套统一的、可度量的标准,为后续的自动化、精细化调度奠定了数学基础,提升了资源分配的公平性和效率。

[0054] 进一步,综合参数p通过以下公式确定:

$$p=\frac{\sum_{t_3=0}\frac{(-1)^{t_3-1}}{t_3}\sum_{t_2=0}\frac{1}{t_3 2^{t_2}+1}\int_0^{t_1^2}\frac{\pi(\sqrt[4]{1+a}-1)\sin a^4}{\sum_{t_3}^{\infty}\frac{((t_3-1)!)^2(2t_1)^{2t_3}}{(2t_3)!}\int_0^1\frac{(1-2t_1)In(1-t_1)}{t_1^2-t_1+1}dt_1}dt_1}{t_1^2(t_1-\tan t_1)In(t_1^2-1)[(\frac{2\arctan\frac{t_4}{t_1}}{\pi})^{t_4}-1]}$$

[0055] 式中,t_1为计算资源,t_2为网络资源,t_3为存储资源,t_4为所需docker附属资源,a为根据系统实际使用设定的参数,用来协调系统对于资源规模的认定,其中t1,t2,t3,t4为正整数,-1<=a<=1。

[0056] 进一步,极值x和y分别通过以下公式确定:

$$x=\lim_{x\to+\infty}\lim_{y\to0}p$$

[0057] $$y=\lim_{x\to0}\lim_{y\to+\infty}p$$

[0058] 式中,P为综合参数。

[0059] 根据上述技术手段,为上述量化参数的量化与分级模型提供了完整、具体的数学定义。通过明确的公式(尽管复杂)固定了资源衡量的算法,使得“统一管理”不再是概念上的,而是具有确定性和可重复性的技术方案,增强了发明的可实现性和保护范围的具体性,防止了因描述模糊而被轻易绕过。

[0060] 进一步,上述方法还包括:建立统一的主机资源管理模型;基于主机资源管理模型评估主机的负载状态,从统一的可扩展资源池中筛选出负载值r小于预设阈值的主机,构成待分配主机集合,其中,负载值r通过以下公式确定:

$$r=\frac{s_2}{s_1}a_1+\frac{s_3}{s_1}a_2+\frac{s_2+s_3}{s_1}a_3$$

[0061] 式中,s_1为主机资源,s_2为已用资源,s_3为资源所用存储;a_1,a_2,a_3为三个0~1之间的调节参数分别视主机剩余资源,待分配目标资源,整体占用资源的重要程度设定。

[0062] 主机资源管理模型是一种专门针对集群中每个物理主机或虚拟机进行状态描述与评估的模型。它通过量化的指标综合评估主机的实时负载、剩余可用资源以及健康状况,用于在调度时判断主机是否适合承接新的工作负载。例如:通过公式计算主机负载值r。其中s1是主机总资源,s2是已用CPU/内存,s3是已用存储。系统设定阈值(如0.8),仅将r<0.8的主机纳入“待分配主机集合”,确保新应用不会被调度到过载节点。

[0063] 根据上述技术手段,引入了针对主机粒度的统一管理模型和量化筛选机制。通过综合考虑主机剩余资源、目标资源需求和整体负载,公式r能够更智能、更全面地评估主机状态,避免将新应用调度到已过载或存储紧张的主机上,从而提升了集群整体的稳定性和

性能。

[0064] 在步骤S202的技术方案中,于Kubernetes将包含多种异构芯片架构的主机集群化,形成一个统一的可扩展资源池。

[0065] 在本实施例中,统一的可扩展资源池指通过容器编排工具(如Kubernetes)将底层所有异构的物理服务器或虚拟机整合起来,在逻辑上形成的一个单一、庞大的资源集合。这个池子对外隐藏了内部节点的架构差异,并可以随着节点的增删而弹性伸缩。

[0066] 例如:一个Kubernetes集群,它纳管了10台X86服务器、5台ARM服务器和3台LoongArch服务器。对平台用户或上层调度器而言,它们看到的是一个拥有18个节点、总计算能力为XXX核的资源池,而不需要关心节点间的架构区别。

[0067] 在步骤S203的技术方案中,接收应用创建请求,基于统一的容器集群模型和基础设施资源管理模型,生成与目标主机的芯片架构无关的应用配置信息。

[0068] 在本实施例中,与目标主机的芯片架构无关的应用配置信息指在应用部署描述文件中,不硬编码任何特定于CPU指令集的依赖(如特定架构的二进制文件、基础镜像标签),仅声明应用的功能性需求(如需要多少计算资源、暴露哪个端口、挂载哪些存储卷)。这份配置可以在任何支持的架构上被解释和执行。

[0069] 例如:一份Kubernetes的YAML部署文件,其中指定了容器镜像为 myapp:latest(该镜像是一个多架构镜像清单),资源请求为 cpu: 500m, memory: 512Mi。这份配置本身不指定myapp:latest具体是x86还是ARM版本,平台会在部署时根据节点架构自动选择正确的镜像层。

[0070] 在步骤S204的技术方案中,根据目标主机的芯片架构信息,通过编程语言的多态性动态调用对应的实现方法,生成与目标主机的芯片架构匹配的容器镜像。

[0071] 在本实施例中,多态性是指一种面向对象的编程特性。在本方案的软件实现中,特指通过定义统一的资源操作接口(基类/接口),然后为每种异构芯片架构提供具体的实现类(派生类)。系统在运行时根据目标主机架构,动态调用对应架构的实现方法,从而生成匹配的配置与镜像。例如:平台有一个统一的“镜像生成器(ImageBuilder)”接口,其中定义了buildImage()方法。针对X86架构有X86ImageBuilder类,针对ARM架构有ARMImageBuilder类,它们分别实现了如何为各自架构构建容器镜像。当调度确定目标主机为ARM时,系统自动调用ARMImageBuilder.buildImage()来生成ARM兼容的镜像。

[0072] 在具体实践中,平台根据获取的目标主机设备信息,动态选择并调用对应的具体实现逻辑,生成与该主机芯片架构匹配的服务配置信息与容器镜像文件。容器镜像的生成与运行基于Docker,平台会使用相应的Docker运行时进行底层设备的轻量级虚拟化及拉取安装与目标架构匹配的依赖镜像。

[0073] 在步骤S205的技术方案中,在统一的可扩展资源池中,基于所生成的容器镜像,创建并运行容器应用。

[0074] 作为一种可选的实施例,在生成与所述目标主机的芯片架构匹配的容器镜像前,该方法包括:基于一个与芯片性能相关的函数F(x)来计算一个优先级参数λ,其中上述计算步骤用以下公式表示:

$$F(x)=4n(\frac{\pi}{4}-\sum_{1}^{x}\frac{n}{n^2+x^2})$$

[0075] $\lambda = b_1 F(x) + b_2$

[0076] 式中x的取值是n,b1,b2用来调节弹性伸缩等对于资源的影响;将优先级参数λ与不同的芯片架构映射;根据优先级参数λ值对待分配主机集合中的主机进行优先级排序,并选择优先级最高的主机用于创建容器应用。

[0077] 根据上述技术手段,在主机筛选的基础上,增加了基于芯片架构性能的智能选择策略。通过函数F(x)和参数λ,系统能够识别不同架构芯片的性能特性差异,并结合弹性伸缩等需求,优先将任务调度到最合适的芯片架构上,实现了从“能用”到“优用”的跨越,进一步优化了应用性能和资源效能。

[0078] 进一步,统一的容器集群模型在进行扩展时,遵循只增加属性或方法的原则,以保证对系统的向后兼容性;在运行容器应用过程中,使用Docker提供底层容器虚拟化,与目标主机的芯片架构匹配的容器镜像由Docker拉取和运行。

[0079] 在具体实践中,容器云平台使用Docker提供底层容器虚拟化,按集群统一管理计算、存储、网络、内存等资源,并通过统一的集群模型对外提供服务。同时,使用Kubernetes提供容器编排服务,从而达到基于异构芯片的容器云平台统一管控的目的。该方法能够充分利用用户在不同时期采购的异构计算资源,极大节约用户的建设成本与运维成本。

[0080] 根据上述技术手段,明确底层依赖(Docker)确保了方案的通用性和可实施性;规定模型的扩展原则(只增不改)则保证了该统一管理平台具有良好的演进性和生态兼容性。当未来出现新的芯片架构时,可以平滑接入而不影响现有系统,极大地延长了平台的生命周期。

[0081] 需要说明的是,对于前述的各方法实施例,为了简单描述,故将其都表述为一系列的动作组合,但是本领域技术人员应该知悉,本申请并不受所描述的动作顺序的限制,因为依据本申请,某些步骤可以采用其他顺序或者同时进行。其次,本领域技术人员也应该知悉,说明书中所描述的实施例均属于优选实施例,所涉及的动作和模块并不一定是本申请所必须的。

[0082] 通过以上的实施方式的描述,本领域的技术人员可以清楚地了解到根据上述实施例的方法可借助软件加必需的通用硬件平台的方式来实现,当然也可以通过硬件,但很多情况下前者是更佳的实施方式。基于这样的理解,本申请的技术方案本质上或者说对现有技术做出贡献的部分可以以软件产品的形式体现出来,该计算机软件产品存储在一个存储介质(如ROM(Read-Only Memory,只读存储器)/RAM(Random Access Memory,随机存取存储器)、磁碟、光盘)中,包括若干指令用以使得一台终端设备(可以是手机,计算机,服务器,或者网络设备等)执行本申请各个实施例的方法。

[0083] 根据本申请实施例的另一个方面,还提供了一种用于实施上述异构芯片容器云平台的统一管理方法的异构芯片容器云平台的统一管理装置。请参阅图3,图3是本申请实施例提供的一种可选的异构芯片容器云平台的统一管理装置的结构框图,如图3所示,该异构芯片容器云平台的统一管理装置300可以包括:

模型管理模块301,被配置为建立统一的容器集群模型和基础设施资源管理模型,以描述异构芯片平台的共性资源与服务;

集群化模块302,被配置为基于Kubernetes将包含多种异构芯片架构的主机集群化,形成一个统一的可扩展资源池;

应用调度模块303,被配置为接收应用创建请求,基于统一的容器集群模型和基础设施资源管理模型,生成与目标主机的芯片架构无关的应用配置信息;

镜像构建模块304,被配置为根据目标主机的芯片架构信息,通过编程语言的多态性动态调用对应的实现方法,生成与目标主机的芯片架构匹配的容器镜像;

容器运行时模块305,被配置为在统一的可扩展资源池中,基于所生成的容器镜像,创建并运行容器应用。

[0084] 需要说明的是,该实施例中的模型管理模块301可以用于执行上述步骤S201,该实施例中的集群化模块302可以用于执行上述步骤S202,该实施例中的应用调度模块303可以用于执行上述步骤S203,该实施例中的镜像构建模块304可以用于执行上述步骤S204,该实施例中的容器运行时模块305可以用于执行上述步骤S205。

[0085] 在本实施例中,模型管理模块301中的基础设施资源管理模型,被配置为通过综合参数p来衡量一个应用所需的资源量,并根据预设的极值x和y,将资源需求划分为n个级别l,其中级别l的计算公式为:$l=\left\lfloor \frac{y-p}{y-x} n \right\rfloor$式中,l为系统资源级别,为1到n之间的整数。

[0086] 综合参数p通过以下公式确定:

$$p=\frac{\sum_{t_3=0}\frac{(-1)^{t_3-1}}{t_3}\sum_{t_2=0}\frac{1}{t_3 2^{t_2}+1}\int_0^{t_1^2}\frac{\pi(\sqrt[4]{1+a}-1)\sin a^4}{\sum_{t_3}^{\infty}\frac{((t_3-1)!)^2(2t_1)^{2t_3}}{(2t_3)!}\int_0^1\frac{(1-2t_1)In(1-t_1)}{t_1^2-t_1+1}dt_1}dt_1}{t_1^2(t_1-\tan t_1)In(t_1^2-1)[(\frac{2\arctan\frac{t_4}{t_1}}{\pi})^{t_4}-1]}$$

[0087] 式中,t_1为计算资源,t_2为网络资源,t_3为存储资源,t_4为所需docker附属资源设,a为根据系统实际使用设定的参数,用来协调系统对于资源规模的认定,其中t1,t2,t3,t4为正整数,-1<=a<=1。极值x和y分别通过以下公式确定:

$$x=\lim_{x\to+\infty}\lim_{y\to 0}p$$

[0088] $$y=\lim_{x\to 0}\lim_{y\to+\infty}p$$

[0089] 式中,P为综合参数。

[0090] 模型管理模块301还用于建立统一的主机资源管理模型;容器运行时模块305包括主机筛选单元,用于基于主机资源管理模型评估主机的负载状态,从统一的可扩展资源池中筛选出负载值r小于预设阈值的主机,构成待分配主机集合,其中,负载值r通过以下公式确定:

$$r=\frac{s_2}{s_1}a_1+\frac{s_3}{s_1}a_2+\frac{s_2+s_3}{s_1}a_3$$

[0091] 式中,s_1为主机资源,s_2为已用资源,s_3为资源所用存储;a_1,a_2,a_3为三个0~1之间的调节参数分别视主机剩余资源,待分配目标资源,整体占用资源的重要程度设定。

[0092] 容器运行时模块305还包括资源选择单元,用于基于一个与芯片性能相关的函数F(x)来计算一个优先级参数λ,其中上述计算步骤用以下公式表示:

$$F(x)=4n(\frac{\pi}{4}-\sum_{1}^{x}\frac{n}{n^2+x^2})$$

[0093] $$\lambda=b_1F(x)+b_2$$

[0094] 式中x的取值是n,b1,b2用来调节弹性伸缩等对于资源的影响;将优先级参数λ与不同的芯片架构映射;根据优先级参数λ值对待分配主机集合中的主机进行优先级排序,并选择优先级最高的主机用于创建容器应用。

[0095] 镜像构建模块304和容器运行时模块305基于Docker提供底层容器虚拟化;模型管理模块被配置为在对统一模型进行扩展时,遵循只增加属性或方法的原则,以保证向后兼容性。

[0096] 关于本实施例中的异构芯片容器云平台的统一管理装置,其模型管理模块301、集群化模块302、应用调度模块303、镜像构建模块304以及容器运行时模块305执行上述异构芯片容器云平台的统一管理方法的具体方式已经在有关该异构芯片容器云平台的统一管理方法的实施例中进行了详细描述,此处将不作详细阐述说明。

[0097] 可以理解的是,本实施例提供的技术方案,该异构芯片容器云平台的统一管理装置中各个模块,通过构建架构无关的统一抽象层(模型与资源池),并在运行时动态绑定具体架构实现(多态性生成镜像),从根本上解决了ARM、MIPS、LoongArch、x86等异构芯片无法在同一容器云平台下统一管理、调度和运维的业界难题,实现了资源池化、管理标准化和运维简化,显著提高了资源利用率和降低了采购与运维成本。

[0098] 本实施例中的装置,除包含上述模块之外,还可以包含执行如前述任一异构芯片容器云平台的统一管理方法的实施例中任意方法的模块。

[0099] 此处需要说明的是,上述模块与对应的步骤所实现的示例和应用场景相同,但不限于上述实施例所公开的内容。需要说明的是,上述模块作为装置的一部分可以运行在实现如图1所示方法的硬件环境中,可以通过软件实现,也可以通过硬件实现,其中,硬件环境包括网络环境。

[0100] 根据本申请实施例的又一个方面,还提供了一种用于实施上述异构芯片容器云平台的统一管理方法的电子设备,该电子设备可以是服务器、终端、或者其组合。

[0101] 根据本申请的另一个实施例,还提供一种电子设备,请参阅图4,图4是本申请实施例提供的一种可选的电子设备的结构框图,如图4所示,该电子设备可以包括:处理器1501、通信接口1502、存储器1503和通信总线1504,其中,处理器1501,通信接口1502,存储器1503通过通信总线1504完成相互间的通信。

[0102] 存储器1503,用于存放计算机程序;

处理器1501,用于执行存储器1503上所存放的程序时,实现如下步骤:

步骤S201,建立统一的容器集群模型和基础设施资源管理模型,以描述异构芯片平台的共性资源与服务;

步骤S202,基于Kubernetes将包含多种异构芯片架构的主机集群化,形成一个统一的可扩展资源池;

步骤S203,接收应用创建请求,基于统一的容器集群模型和基础设施资源管理模型,生成与目标主机的芯片架构无关的应用配置信息;

步骤S204,根据目标主机的芯片架构信息,通过编程语言的多态性动态调用对应

的实现方法,生成与目标主机的芯片架构匹配的容器镜像;

步骤S205,在统一的可扩展资源池中,基于所生成的容器镜像,创建并运行容器应用。

[0103] 可以理解的是,本实施例提供的技术方案,该电子设备的处理器,通过构建架构无关的统一抽象层(模型与资源池),并在运行时动态绑定具体架构实现(多态性生成镜像),从根本上解决了ARM、MIPS、LoongArch、x86等异构芯片无法在同一容器云平台下统一管理、调度和运维的业界难题,实现了资源池化、管理标准化和运维简化,显著提高了资源利用率和降低了采购与运维成本。

[0104] 可选地,在本实施例中,上述的通信总线可以是PCI (Peripheral Component Interconnect,外设部件互连标准)总线、或EISA (Extended Industry Standard Architecture,扩展工业标准结构)总线等。该通信总线可以分为地址总线、数据总线、控制总线等。为便于表示,图中仅用一条粗线表示,但并不表示仅有一根总线或一种类型的总线。通信接口用于上述电子设备与其他设备之间的通信。

[0105] 存储器可以包括随机存取存储器(Random Access Memory,RAM),也可以包括非易失性存储器(Non-Volatile Memory,NVM),例如至少一个磁盘存储器。可选的,存储器还可以是至少一个位于远离前述处理器的存储装置。

[0106] 上述处理器可以是通用处理器,可以包含但不限于:CPU (Central Processing Unit,中央处理器)、NP(Network Processor,网络处理器)等;还可以是DSP (Digital Signal Processor,数字信号处理器)、ASIC (Application Specific Integrated Circuit,专用集成电路)、FPGA (Field—Programmable Gate Array,现场可编程门阵列)或者其他可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件。

[0107] 本申请实施例还提供一种计算机可读存储介质,存储介质包括存储的程序,其中,程序运行时执行上述方法实施例的方法步骤。

[0108] 可选地,在本实施例中,上述存储介质可以包括但不限于:U盘、ROM、RAM、移动硬盘、磁碟或者光盘等各种可以存储程序代码的介质。

[0109] 上述本申请实施例序号仅仅为了描述,不代表实施例的优劣。

[0110] 上述实施例中的集成的单元如果以软件功能单元的形式实现并作为独立的产品销售或使用时,可以存储在上述计算机可读取的存储介质中。基于这样的理解,本申请的技术方案本质上或者说对现有技术做出贡献的部分或者该技术方案的全部或部分可以以软件产品的形式体现出来,该计算机软件产品存储在存储介质中,包括若干指令用以使得一台或多台计算机设备(可为个人计算机、服务器或者网络设备等)执行本申请各个实施例方法的全部或部分步骤。

[0111] 在本申请的上述实施例中,对各个实施例的描述都各有侧重,某个实施例中没有详述的部分,可以参见其他实施例的相关描述。

[0112] 在本申请所提供的几个实施例中,应该理解到,所揭露的客户端,可通过其它的方式实现。其中,以上所描述的装置实施例仅仅是示意性的,例如单元的划分,仅仅为一种逻辑功能划分,实际实现时可以有另外的划分方式,例如多个单元或组件可以结合或者可以集成到另一个系统,或一些特征可以忽略,或不执行。另一点,所显示或讨论的相互之间的耦合或直接耦合或通信连接可以是通过一些接口,单元或模块的间接耦合或通信连接,可

以是电性或其它的形式。

[0113] 作为分离部件说明的单元可以是或者也可以不是物理上分开的,作为单元显示的部件可以是或者也可以不是物理单元,即可以位于一个地方,也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例中所提供的方案的目的。

[0114] 另外,在本申请各个实施例中的各功能单元可以集成在一个处理单元中,也可以是各个单元单独物理存在,也可以两个或两个以上单元集成在一个单元中。上述集成的单元既可以采用硬件的形式实现,也可以采用软件功能单元的形式实现。

[0115] 以上仅是本申请的优选实施方式,应当指出,对于本技术领域的普通技术人员来说,在不脱离本申请原理的前提下,还可以做出若干改进和润饰,这些改进和润饰也应视为本申请的保护范围。

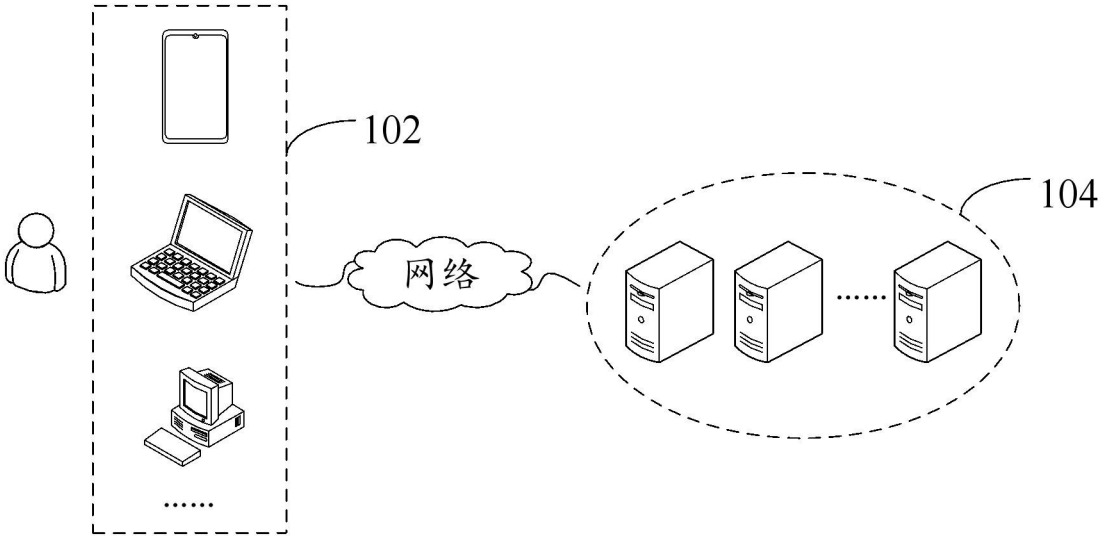

102
104
网络
.....
.....

图1

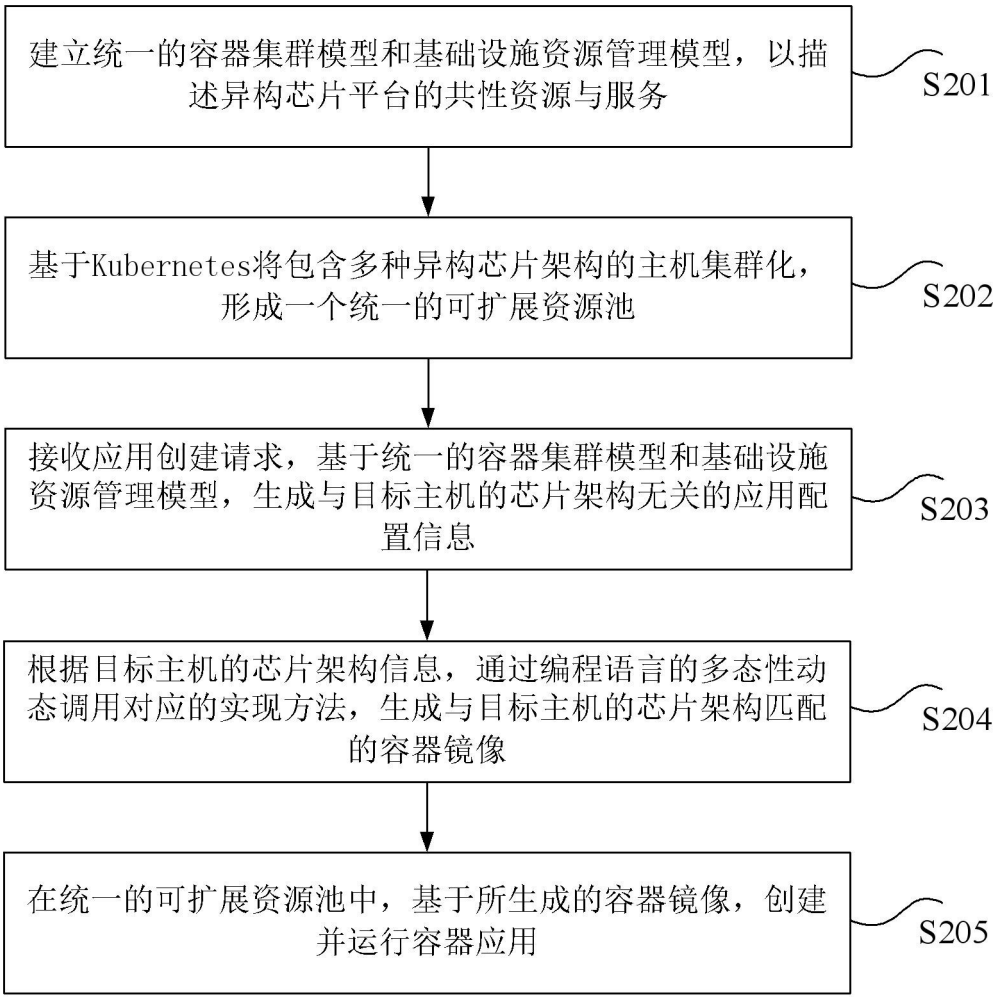

建立统一的容器集群模型和基础设施资源管理模型，以描述异构芯片平台的共性资源与服务
S201
基于Kubernetes将包含多种异构芯片架构的主机集群化，形成一个统一的可扩展资源池
S202
接收应用创建请求，基于统一的容器集群模型和基础设施资源管理模型，生成与目标主机的芯片架构无关的应用配置信息
S203
根据目标主机的芯片架构信息，通过编程语言的多态性动态调用对应的实现方法，生成与目标主机的芯片架构匹配的容器镜像
S204
在统一的可扩展资源池中，基于所生成的容器镜像，创建并运行容器应用
S205

图2

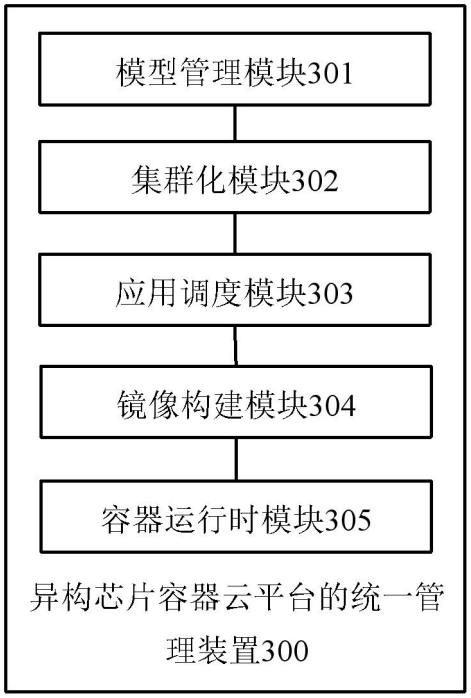

模型管理模块301
集群化模块302
应用调度模块303
镜像构建模块304
容器运行时模块305
异构芯片容器云平台的统一管理装置300

图3

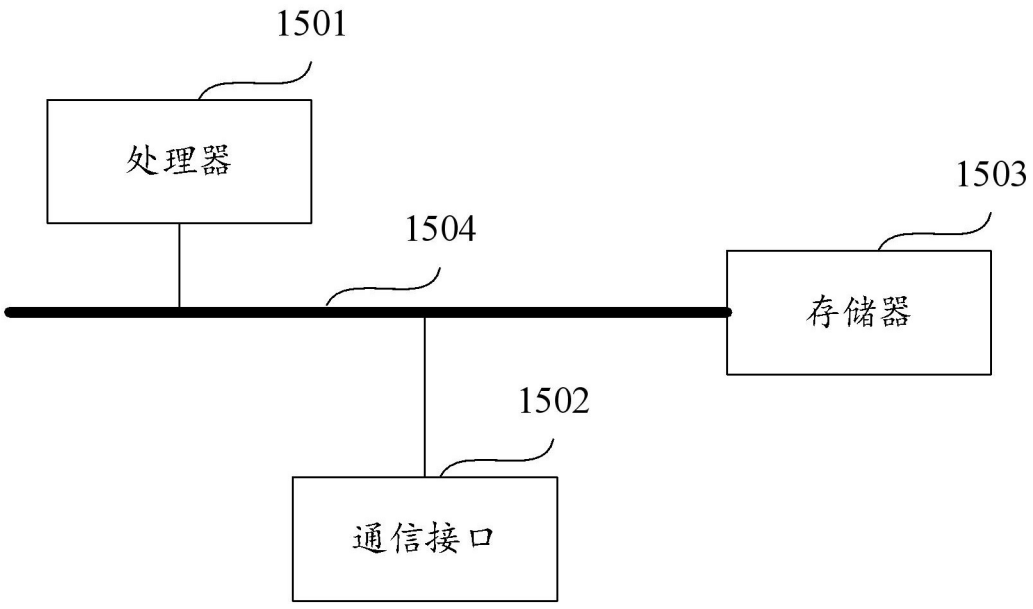

1501
处理器
1503
1504
存储器
1502
通信接口

图4

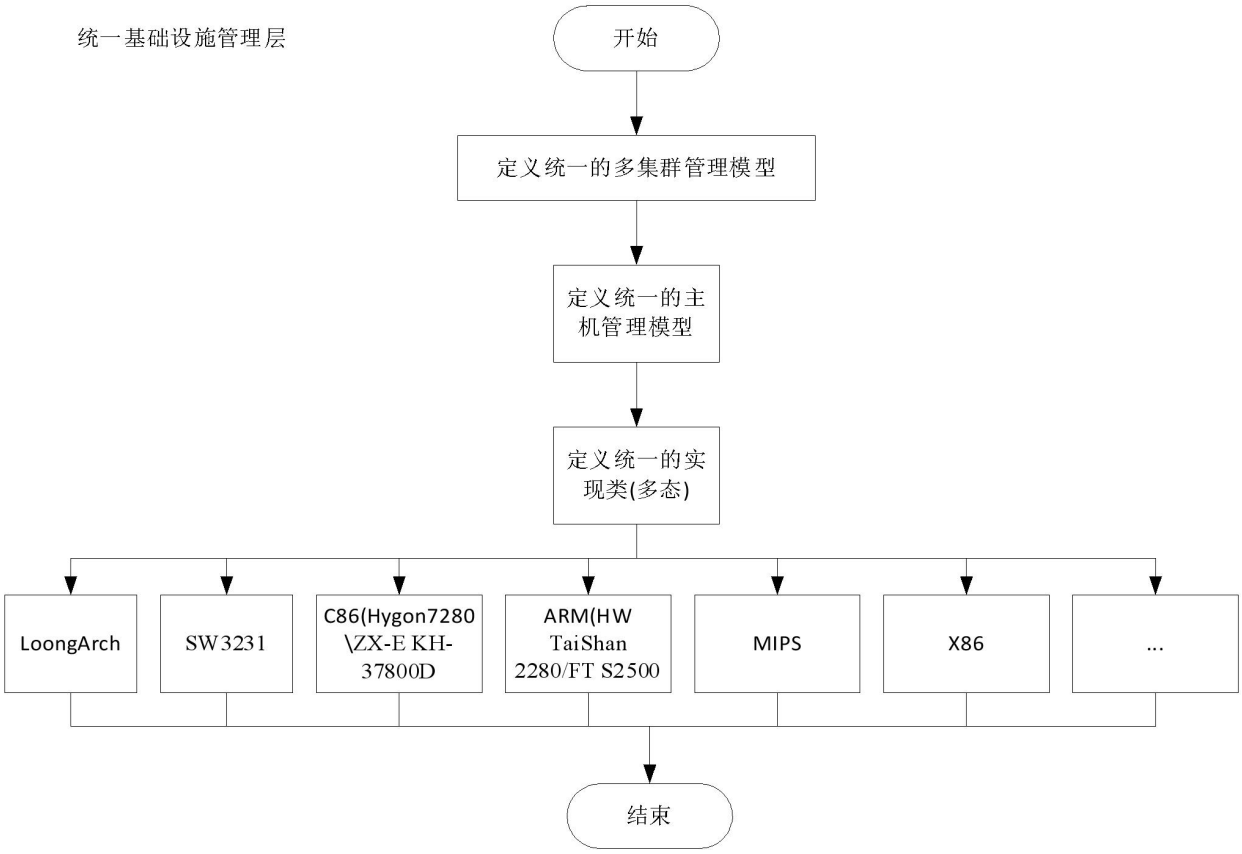

统一基础设施管理层
开始
定义统一的多集群管理模型
定义统一的主机管理模型
定义统一的实现类(多态)
LoongArch
SW3231
C86(Hygon7280\ZX-E KH-37800D
ARM(HW TaiShan 2280/FT S2500
MIPS
X86
...
结束

图5