(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 119045955 B
(45)授权公告日 2025.03.28

(21)申请号 202411545067.3
(22)申请日 2024.10.31
(65)同一申请的已公布的文献号
申请公布号 CN 119045955 A
(43)申请公布日 2024.11.29
(73)专利权人 阿里云计算有限公司
地址 310024 浙江省杭州市西湖区三墩镇灯彩街1008号云谷园区1-2-A06室
(72)发明人 胡璧涛 刘嵩 徐誉畅 杨勇 马涛
(74)专利代理机构 北京太合九思知识产权代理有限公司 11610
专利代理师 刘瑞霞
(51)Int.Cl.
G06F 9/455(2018.01)

(56)对比文件
CN 112463297 A,2021.03.09
US 2016077846 A1,2016.03.17

审查员 倪霞

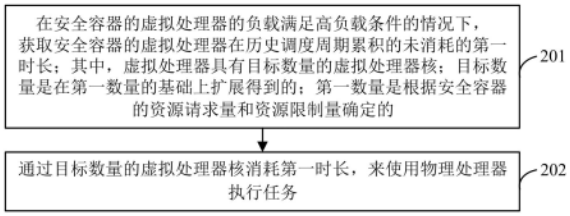
权利要求书3页 说明书17页 附图4页

(54)发明名称

处理器时间调度和容器创建方法、设备、介质及程序产品

(57)摘要

本申请实施例提供一种处理器时间调度和容器创建方法、设备、介质及程序产品。在本申请实施例中,为了在安全容器内实现弹性处理器带宽控制,通过对传统安全容器分配到的虚拟处理器核进行扩展,打破了传统安全容器中限制使用时长(Quota)与虚拟处理器核需要一比一的限制,安全容器可在高负载时,利用扩展出的虚拟处理器核,消耗历史调度周期累积的未消耗的时长,实现弹性处理器带宽控制,有助于满足突发型负载需求,提高安全容器的服务质量。



CN 119045955 B

1.一种处理器时间调度方法,其特征在于,包括:

在安全容器的虚拟处理器的负载满足高负载条件的情况下,获取所述安全容器的虚拟处理器在历史调度周期累积的未消耗的第一时长;所述虚拟处理器具有目标数量的虚拟处理器核;所述目标数量是在创建所述安全容器时在第一数量的基础上扩展得到的;所述第一数量是在创建所述安全容器时根据所述安全容器的资源请求量和资源限制量确定的;所述虚拟处理器在任一历史调度周期未消耗的时长,等于安全容器的限制使用时长减去所述目标数量的虚拟处理器核完成所述任一历史调度周期的任务所消耗的总时间;所述限制使用时长等于所述第一数量与调度周期的乘积;

通过所述目标数量的虚拟处理器核消耗所述第一时长,来使用物理处理器执行任务。

2.根据权利要求1所述的方法,其特征在于,所述通过所述目标数量的虚拟处理器核消耗所述第一时长,来使用物理处理器执行任务,包括:

针对当前调度周期,从所述第一时长中获取所述安全容器在当前调度周期可超限使用物理处理器的第二时长;

在所述安全容器对应的限制使用时长的基础上增加所述第二时长,以得到目标时长;所述限制使用时长是指所述安全容器使用物理处理器的时间上限;

为所述目标数量的虚拟处理器核共分配所述目标时长的时间,以供所述目标数量的虚拟处理器核在分配到的时间内使用所述物理处理器执行当前调度周期的任务。

3.根据权利要求2所述的方法,其特征在于,所述从所述第一时长中获取所述安全容器在当前调度周期可超限使用物理处理器的第二时长,包括:

计算扩展倍数与所述限制使用时长的乘积,作为所述虚拟处理器在当前调度周期可使用所述物理处理器的总时长;所述扩展倍数等于所述目标数量除以所述第一数量;

计算所述总时长与所述限制使用时长之间的第一差值,作为所述第二时长,并从所述第一时长中获取所述第二时长。

4.根据权利要求2所述的方法,其特征在于,所述安全容器所在的宿主机上部署有其它容器,所述安全容器与所述其它容器共享所述物理处理器;所述方法还包括:

在安全容器的虚拟处理器的负载满足设定的高负载条件的情况下,降低所述安全容器的调度权重,以得到所述安全容器的目标调度权重;

根据所述其它容器的调度权重及所述安全容器的目标调度权重,确定所述安全容器在当前调度周期使用所述物理处理器的第三时长;所述目标调度权重小于所述其它容器的调度权重;

若所述第三时长大于所述安全容器对应的限制使用时长,则根据所述第三时长与所述限制使用时长之间的第二差值,确定所述第二时长。

5.根据权利要求4所述的方法,其特征在于,所述方法还包括:

计算扩展倍数及所述限制使用时长的乘积,作为所述安全容器在当前调度周期可使用所述物理处理器的总时长;所述扩展倍数等于所述目标数量除以所述第一数量;

计算所述总时长与所述限制使用时长之间的第一差值;

所述根据所述第三时长与所述限制使用时长之间的第二差值,确定所述第二时长,包括:

选择所述第一差值和所述第二差值中最小值,作为所述第二时长。

6.根据权利要求2所述的方法,其特征在于,所述为所述目标数量的虚拟处理器核共分配所述目标时长的时间,包括:

以所述目标数量的虚拟处理器核分配到的时间之和等于所述目标时长为约束条件,根据各虚拟处理器核在当前调度周期执行的任务的资源需求,确定所述目标数量的虚拟处理器核各自对应的待分配时长;

为各虚拟处理器核分配对应的待分配时长的时间,来为所述目标数量的虚拟处理器核共分配目标时长的时间。

7.根据权利要求1-6任一项所述的方法,其特征在于,所述方法还包括:

针对当前调度周期的前一调度周期,若所述目标数量的虚拟处理器核在所述前一调度周期完成任务所消耗的总时间达到所述安全容器对应的限制使用时长,则确定所述安全容器的虚拟处理器的负载满足设定的高负载条件。

8.根据权利要求1-6任一项所述的方法,其特征在于,所述方法还包括:

获取容器创建请求,所述容器创建请求用于请求创建所述安全容器,所述容器创建请求包括所述资源请求量和所述资源限制量;

根据所述资源请求量和所述资源限制量,确定所述虚拟处理器的虚拟处理器核数,作为所述第一数量;

对所述第一数量进行扩展,以得到所述目标数量;

根据所述目标数量的虚拟处理器核,在宿主机上创建虚拟机实例;并在所述虚拟机实例上创建所述安全容器,以使所述安全容器的虚拟处理器具有所述目标数量的虚拟处理器核。

9.根据权利要求8所述的方法,其特征在于,所述容器创建请求还包括:扩展倍数;所述扩展倍数大于1;

所述对所述第一数量进行扩展,以得到所述目标数量,包括:

根据所述扩展倍数对所述第一数量进行扩展,以得到所述目标数量。

10.根据权利要求8所述的方法,其特征在于,还包括:

响应于所述安全容器对处理器信息的访问操作,向所述安全容器提供所述第一数量的虚拟处理器核的资源视图。

11.根据权利要求10所述的方法,其特征在于,所述响应于所述安全容器对处理器信息的访问操作,向所述安全容器提供所述第一数量的虚拟处理器核的资源视图,包括:

响应于所述安全容器对处理器信息的访问操作,确定所述目标数量的虚拟处理器核对应的第一位图;所述第一位图的一个比特位对应一个虚拟处理器核;

根据扩展倍数,将所述第一位图修改为第二位图;所述第二位图中目标比特位的数量等于所述第一数量;所述目标比特位是指表征对应有虚拟处理器核的比特位;所述扩展倍数等于所述目标数量除以所述第一数量;

将所述第二位图提供给所述安全容器,以向所述安全容器提供所述第一数量的虚拟处理器核的资源视图。

12.根据权利要求11所述的方法,其特征在于,还包括:

响应于所述安全容器的处理器绑定操作,获取所述处理器绑定操作待绑定的目标虚拟处理器核及待绑定的进程;

根据所述扩展倍数,确定所述目标虚拟处理器核对应的目标比特位在所述第一位图中对应的所述扩展倍数个比特位;

将所述待绑定的进程与所述扩展倍数个比特位对应的虚拟处理器核进行绑定。

13.一种容器创建方法,其特征在于,包括:

获取容器创建请求,所述容器创建请求用于请求创建安全容器,所述容器创建请求包括资源请求量和资源限制量;

根据所述资源请求量和所述资源限制量,确定所述安全容器的虚拟处理器核数,作为第一数量;

对所述第一数量进行扩展,以得到目标数量;

根据所述目标数量的虚拟处理器核,在宿主机上创建虚拟机实例;

在所述虚拟机实例上创建所述安全容器,以在所述安全容器的虚拟处理器的负载满足高负载条件的情况下,利用所述目标数量的虚拟处理器核,消耗所述安全容器之前累积的未消耗的处理器时长来执行任务;

其中,所述安全容器在任一历史调度周期未消耗的处理器时长,等于安全容器的限制使用时长减去所述目标数量的虚拟处理器核完成所述任一历史调度周期的任务所消耗的总时间;所述限制使用时长等于所述第一数量与调度周期的乘积。

14.一种电子设备,其特征在于,包括:存储器和处理器;其中,所述存储器,用于存储计算机程序;

所述处理器耦合至所述存储器,用于执行所述计算机程序以用于执行权利要求1-13任一项所述方法中的步骤。

15.一种存储有计算机指令的计算机可读存储介质,其特征在于,当所述计算机指令被一个或多个处理器执行时,致使所述一个或多个处理器执行权利要求1-13任一项所述方法中的步骤。

16.一种计算机程序产品,其特征在于,包括计算机程序,当所述计算机程序被一个或多个处理器执行时,致使所述一个或多个处理器执行权利要求1-13任一项所述方法中的步骤。

处理器时间调度和容器创建方法、设备、介质及程序产品

技术领域

[0001] 本申请涉及计算机技术领域,尤其涉及一种处理器时间调度和容器创建方法、设备、介质及程序产品。

背景技术

[0002] 随着云计算技术的发展,容器技术在实现多租户环境中扮演了重要角色。容器技术允许在同一台宿主机上运行多个相互隔离的应用程序实例,从而提高了资源利用率和部署效率。

[0003] 在处理中央处理器(Central Processing Unit,CPU)突发型任务时,传统共享型容器可通过弹性CPU带宽控制技术提供较好的服务质量,但在安全性方面存在较大隐患。为了在保障安全性的前提下继续提供高质量的服务,安全容器被广泛使用。然而现有安全容器,无法使用弹性CPU带宽控制技术。

发明内容

[0004] 本申请的多个方面提供一种处理器时间调度和容器创建方法、设备、介质及程序产品,用以在安全容器中实现弹性处理器带宽控制。

[0005] 本申请实施例提供一种处理器时间调度方法,包括:

[0006] 在安全容器的虚拟处理器的负载满足高负载条件的情况下,获取所述安全容器的虚拟处理器在历史调度周期累积的未消耗的第一时长;所述虚拟处理器具有目标数量的虚拟处理器核;所述目标数量是在所述第一数量的基础上扩展得到的;所述第一数量是根据所述安全容器的资源请求量和资源限制量确定的;

[0007] 通过所述目标数量的虚拟处理器核消耗所述第一时长,来使用物理处理器执行任务。

[0008] 本申请实施例还提供一种容器创建方法,包括:

[0009] 获取容器创建请求,所述容器创建请求用于请求创建安全容器,所述容器创建请求包括资源请求量和所述资源限制量;

[0010] 根据所述资源请求量和所述资源限制量,确定所述安全容器的虚拟处理器核数,作为第一数量;

[0011] 对所述第一数量进行扩展,以得到目标数量;

[0012] 根据所述目标数量的虚拟处理器核,在宿主机上创建虚拟机实例;

[0013] 在所述虚拟机实例上创建所述安全容器,以在所述安全容器的虚拟处理器的负载满足高负载条件的情况下,利用所述目标数量的虚拟处理器核,消耗所述安全容器之前累积的未消耗的处理器时长来执行任务。

[0014] 本申请实施例还提供一种电子设备,包括:存储器和处理器;其中,所述存储器,用于存储计算机程序;

[0015] 所述处理器耦合至所述存储器,用于执行所述计算机程序以用于执行前述处理器

时间调度方法和/或容器创建方法中的步骤。

[0016] 本申请实施例还提供一种存储有计算机指令的计算机可读存储介质,当所述计算机指令被一个或多个处理器执行时,致使所述一个或多个处理器执行前述处理器时间调度方法和/或容器创建方法中的步骤。

[0017] 本申请实施例还提供一种计算机程序产品,包括计算机程序,当所述计算机程序被一个或多个处理器执行时,致使所述一个或多个处理器执行前述处理器时间调度方法和/或容器创建方法中的步骤。

[0018] 在本申请实施例中,为了在安全容器内实现弹性处理器带宽控制,通过对传统安全容器分配到的虚拟处理器核进行扩展,打破了传统安全容器中限制使用时长(Quota)与虚拟处理器核需要一比一的限制,安全容器可在高负载时,利用扩展出的虚拟处理器核,消耗历史调度周期累积的未消耗的时长,实现弹性处理器带宽控制,有助于满足突发型负载需求,提高安全容器的服务质量。

附图说明

[0019] 此处所说明的附图用来提供对本申请的进一步理解,构成本申请的一部分,本申请的示意性实施例及其说明用于解释本申请,并不构成对本申请的不当限定。在附图中:

[0020] 图1a和图1b为本申请实施例提供的容器创建方法的流程示意图;

[0021] 图2为本申请实施例提供的处理器时间调度方法的流程示意图;

[0022] 图3为本申请实施例提供的安全容器的架构图;

[0023] 图4为本申请实施例提供的扩展出的虚拟处理器核的隐藏机制示意图;

[0024] 图5为本申请实施例提供的扩展出的虚拟处理器核的映射机制示意图;

[0025] 图6为本申请实施例提供的电子设备的结构示意图。

具体实施方式

[0026] 为使本申请的目的、技术方案和优点更加清楚,下面将结合本申请具体实施例及相应的附图对本申请技术方案进行清楚、完整地描述。显然,所描述的实施例仅是本申请一部分实施例,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,都属于本申请保护的范围。

[0027] 需要说明的是,本申请所涉及的用户信息(包括但不限于用户设备信息、用户个人信息等)和数据(包括但不限于用于分析的数据、存储的数据、展示的数据等),均为经用户授权或者经过各方充分授权的信息和数据,并且相关数据的收集、使用和处理需要遵守相关国家和地区的相关法律法规和标准,并提供有相应的操作入口,供用户选择授权或者拒绝。

[0028] 下面对本申请实施例涉及的术语或概念进行解释说明。

[0029] 传统容器:使用资源控制组(Control Group,Cgrou)及命名空间(Namespace)技术实现运行时隔离的容器。特点是容器与宿主机共享内核。其中,Cgroup 是操作系统(Operating System,OS)的内核提供的一种机制,用于对进程组进行资源管理和控制,可限制、记录和隔离进程组使用的物理资源。物理资源可包括处理器资源、内存资源及磁盘输入/输出(Input/Output,I/O)资源等。它允许系统管理员或进程调度器将进程分配到不同

的资源控制组(Cgroups),然后为这些资源控制组设置资源使限制、优先级或记账等。Cgroup是容器技术的核心组件之一。Cgroup 可提高服务器和容器环境中的资源利用率和隔离性,是实现轻量级虚拟化技术的基础之一。

[0030] 安全容器:安全容器是一种增强型的容器技术,在提供比传统容器更高的安全隔离级别。与传统容器相比,安全容器通过引入虚拟化层,提供更强的安全性和隔离性。安全容器通常通过引入轻量级虚拟化技术的隔离机制,将容器的运行环境与宿主机的内核完全隔离。这种隔离确保了容器内的应用程序不会直接影响到宿主机和其他容器的运行,降低了安全漏洞的风险。安全容器使用单独的虚拟化内核,而不是与宿主机共享内核,从而防止了潜在的内核级攻击。每个容器都有自己的虚拟化内核实例,这样即便容器内部发生安全事件,也不会影响到宿主机和其他容器。

[0031] 共享型安全容器:采用非绑定CPU调度模式,每个虚拟CPU(VirtualCPU,vCPU)被随机分配到任何空闲的物理CPU上,不同vCPU会争抢物理CPU资源。

[0032] 弹性CPU带宽控制:通过累积在CPU相对空闲时每个调度周期内未使用完的CPU时间,在CPU高负载期间使用这些累积的时间,从而突破原本设定的处理器配额(Quota)限制,提升服务质量。这种方法是在处理那些平均CPU利用率不高但某些时间段CPU利用率很高的CPU突发型任务时,效果显著。

[0033] CPU配额(CPU Quota):处理器配额,如CPU配额(CPU Quota)策略,用于限制容器在一定周期内可以使用的处理器(如CPU)的时间总量。当容器的限制(Limit)设置了处理器资源时,可在资源控制组(Cgroup)中设置处理器配额策略(如CPU Quota)来限制容器的处理器资源使用。具体地,可将容器的限制(Limit)转换为资源控制组(Cgroup)中的处理器配额策略(如CPU Quota),从而限制容器在一定周期内可以使用处理器的时间。

[0034] 下面对安全容器无法使用弹性带宽控制技术的原因进行解释说明。

[0035] 传统容器在资源管理和调度方面表现出色,但由于传统容器与宿主机共享内核,一旦被恶意攻破,攻击者可以直接逃逸到宿主机内核,进而影响该宿主机上的所有容器,造成严重的安全隐患。因此,对于安全要求较高的服务,目前的主流趋势是向安全容器迁移。安全容器通过引入轻量级虚拟化层,将容器的内核与宿主机内核隔离,实现了容器与宿主机之间的互不影响。这种隔离机制显著提升了安全性,但同时也带来了新的挑战。

[0036] 在安全容器中,虚拟处理器的每个核对应于宿主机上的一个进程,因此容器内运行的任务受限于虚拟处理器核的数量。这意味着容器规格不仅表示该容器可使用的处理器资源上限,还表示该容器的虚拟处理器的核的数量。虚拟处理器的核是指逻辑核。一个物理核可虚拟化为一个或多个逻辑核。多个是指2个以上(含2个)。

[0037] 一些容器集群管理系统,如Kubernetes(K8s)使用处理器(如CPU)请求(request.cpu)和限制(limit.cpu)参数分配与控制容器的CPU资源。请求(request.cpu)代表这个容器分配时间片的权重值,是容器的最小处理器资源保障。请求(request.cpu)指容器期望获得的最小CPU资源量。限制(limit.cpu)表示容器可使用的CPU时间的硬性上限。比如,request.cpu=1,limit.cpu=4表示当宿主机上有超过1份空闲CPU时间的情况下,容器可以得到调度,但该容器可用的CPU时间上限不超过4份CPU时间。创建安全容器时会使用请求(request.cpu)和限制(limit.cpu)这两个参数确定虚拟处理器(vCPU)的核的数量,也可称为虚拟处理器的数量。计算方法如下面公式:

[0038] nr_vcpu = max(request.cpu, limit.cpu)(1)。

[0039] 其中,nr_vcpu表示虚拟处理器的核的数量;max(request.cpu, limit.cpu)表示取请求(request.cpu)和限制(limit.cpu)的最大值。

[0040] 进一步,安全容器的控制面,如运行时(Runtime)组件可依据该计算结果创建虚拟机,虚拟机中处理器的核数等于nr_vcpu,虚拟处理器的每个核对应一个宿主机上的进程。其中,运行时(Runtime)组件负责管理容器的生命周期,包括创建、启动、停止和销毁容器。容器是由容器运行时根据容器镜像创建出来的独立运行环境。

[0041] 然后,安全容器控制面会为该安全容器设置处理器配额(Quota),限制其可使用的处理器(如CPU)时间上限,计算方法如下面公式:

[0042] cpu.quota = nr_vcpu * cpu.period(2)。

[0043] 其中,cpu.quota表示容器可使用的处理器时间上限;cpu.period表示CPU调度周期(Period),等于一份CPU时间。

[0044] 结合上述两个公式可得,安全容器控制面(如运行时组件)在创建安全容器时,直接将处理器配额(Quota)与虚拟处理器的核进行了一对一的映射,这就直接限制了在安全容器的CPU相对空闲时累积的未使用完的CPU时间,无法在CPU忙时消费。因为在安全容器的CPU忙时无多余的虚拟处理器核来消耗之前累加的未使用完的CPU时间,也就导致弹性CPU带宽控制技术无法在安全容器中生效。然而,安全容器的平均CPU利用率不高,而某些时间段CPU利用率很高的CPU突发型负载是普遍存在的。因此,为提升安全容器内CPU突发型负载的服务质量,需要安全容器支持弹性CPU带宽控制技术。

[0045] 在本申请一些实施例中,为了在安全容器内实现弹性处理器带宽控制,通过对传统安全容器分配到的虚拟处理器核进行扩展,打破了传统安全容器中限制使用时长(Quota)与虚拟处理器核需要一比一的限制,安全容器可在高负载时,利用扩展出的虚拟处理器核,消耗历史调度周期累积的未消耗的时长,实现弹性处理器带宽控制,有助于满足突发型负载需求,提高安全容器的服务质量。

[0046] 以下结合附图,详细说明本申请各实施例提供的技术方案。

[0047] 应注意到:相同的标号在下面的附图以及实施例中表示同一物体,因此,一旦某一物体在一个附图或实施例中被定义,则在随后的附图和实施例中不需要对其进行进一步讨论。

[0048] 图1a为本申请实施例提供的容器创建方法的流程示意图。如图1a所示,该方法主要包括:

[0049] 101、获取容器创建请求,容器创建请求用于请求创建安全容器,容器创建请求包括资源请求量和资源限制量。

[0050] 102、根据资源请求量和资源限制量,确定安全容器的虚拟处理器核数,作为第一数量。

[0051] 103、对第一数量进行扩展,以得到目标数量。

[0052] 104、根据目标数量的虚拟处理器核,在宿主机上创建虚拟机实例。

[0053] 105、在虚拟机实例上创建安全容器,以在安全容器的虚拟处理器的负载满足高负载条件的情况下,利用目标数量的虚拟处理器核,消耗安全容器之前累计的未消耗的处理器时长来执行任务。

[0054] 在本申请实施例中,容器创建请求用于请求创建安全容器,包括:资源请求量(Request)和资源限制量(Limit)。其中,资源请求量(Request)是安全容器至少可获得的资源量,是安全容器的最小资源保障。资源请求量(Request)指容器期望获得的最小资源量。调度器会根据安全容器的资源请求量(Request)为其分配足够的资源。安全容器可使用的最大资源信息可使用安全容器的资源限制量(Limit)确定。容器的资源限制量(Limit)是指容器可以使用的最大资源量。容器的资源使用不得超过这个上限。资源请求量包括前述处理器请求(request.cpu);资源限制量包括处理器限制(如limit.cpu)。

[0055] 在一些实施例中,容器创建请求可包括待创建的安全容器的配置文件,如yaml文件。yaml文件包括前述资源请求量和资源限制量。其中,yaml文件提供了一种简洁的方式来描述安全容器的配置数据结构,非常适合用来配置和管理容器。

[0056] 在步骤101中获取容器创建请求之后,在步骤102中,可根据资源请求量和资源限制量,确定安全容器的虚拟处理器核数。具体地,可从处理器请求和处理器限制中,选择最大值,作为安全容器的虚拟处理器核数,即:nr_vcpu = max(request.cpu, limit.cpu)。此处的安全容器的虚拟处理器核数可称为安全容器的标准虚拟处理器核数。

[0057] 在本实施例中,为了使安全容器能够支持处理器弹性控制技术,可对安全容器的标准虚拟处理器核数进行扩展,扩展出的虚拟处理器核数可用于在安全容器处于高负载时,消耗安全容器的处理器相对闲时累加的处理器时长。因此,在步骤103中,可对安全容器的虚拟处理器核数进行扩展,以得到目标数量。其中,目标数量是指扩展后的虚拟处理器核数。在本申请实施例中,为了便于描述和区分,将步骤102确定出的安全容器的标准虚拟处理器核数,定义为第一数量;将对标准虚拟处理器核数扩展出的虚拟处理器核数,定义为第二数量,则目标数量等于第一数量加上第二数量。

[0058] 在本申请实施例中,不限定对第一数量进行扩展的具体实施方式。在一些实施例中,安全容器的用户可指定扩展倍数(burst_vcpu_factor)。相应地,容器创建请求可包括扩展倍数。其中,扩展倍数也可位于安全容器的配置文件,如yaml文件中。即在配置文件中增加扩展倍数对应的参数,用户可通过给该参数赋值,指定扩展倍数。其中,扩展倍数大于或等于1。一般地,扩展倍数为整数。由于扩展倍数等于1时,安全容器为普通的安全容器,无法使用弹性处理器带宽控制技术,因此,本申请实施例重点针对扩展倍数大于1进行说明。基于扩展倍数,步骤103可实现为:根据扩展倍数对第一数量进行扩展,以得到目标数量。具体地,可将第一数量与扩展倍数的乘积,作为目标数量。相应地,目标数量可表示为:

[0059] nr_burst_vcpu = nr_vcpu * burst_vcpu_factor ,burst_vcpu_factor≥1(3)。

[0060] 其中,nr_burst_vcpu表示虚拟处理器核的目标数量;burst_vcpu_factor表示扩展倍数。

[0061] 在该实施例中,可根据安全容器的配置文件中是否携带有扩展倍数对应的参数(如burst_vcpu_factor)来判断是否使用本申请实施例提供的容器创建方法创建安全容器。

[0062] 在另一些实施例中,可在第一数量的基础上增加设定数量,以得到目标数量。或者,可将第一数量乘以设定倍数,得到目标数量。其中,设定倍数大于1。在这些实施例中,可在安全容器的配置文件中增加用于指示是否使用本申请实施例提供的容器创建方法创建安全容器的参数。若该参数的值表征使用本申请实施例提供的容器创建方法创建安全容

器,则采用使用本申请实施例提供的容器创建方法创建安全容器;若该参数的值表征不使用本申请实施例提供的容器创建方法创建安全容器,则采用前述实施例提供的传统安全容器创建方法创建安全容器。

[0063] 进一步,在步骤104中,可根据目标数量的虚拟处理器核,在宿主机上创建虚拟机实例。具体地,可利用基于内核的虚拟机技术(Kernel-based Virtual Machine,KVM)技术在宿主机上创建虚拟机,并为虚拟机配置目标数量的虚拟处理器核及其它资源,以得到虚拟机实例。其中,其它资源包括:内存资源、持久化存储资源及网络资源等,但不限于此。其中,KVM是操作系统(如Linux系统)下的虚拟化技术,可将操作系统转变为虚拟机监测程序,使主机计算机能够运行多个隔离的虚拟环境。

[0064] 进一步,在步骤105中,可在虚拟机实例上创建安全容器,以使安全容器的虚拟处理器具有目标数量的虚拟处理器核。在虚拟机实例上创建安全容器,可为安全容器运行一个独立的内核,以确保安全容器的可靠性。

[0065] 具体地,可在虚拟机实例上加载待创建的安全容器的容器镜像;并根据容器镜像,在虚拟机实例上创建安全容器。具体地,可将容器镜像中的代码运行在KVM创建的虚拟机实例上,进而在虚拟机实例上创建出安全容器,使得安全容器运行在KVM创建的独立内核上。

[0066] 值得说明的是,本申请实施例涉及的安全容器可为共享型安全容器,也可为独占型安全容器。对于共享型安全容器,安全容器内的每个虚拟处理器核是一个进程,并且不绑定物理虚拟处理器核,安全容器可用的处理器时间上限由处理器配额(cpu.quota)限制,因此处理器资源的浪费较低,甚至无处理器资源浪费。独占型安全容器内的每个虚拟处理器核绑定到一个物理虚拟处理器核,本申请实施例提供的处理器时间调度方法若应用于独占型安全容器会引入较高的处理器资源浪费。因此,本申请实施例中的安全容器主要是指共享型安全容器。在本申请一些实施例中,除了从安全容器的配置文件中获取资源请求量、资源限制量及扩展倍数之外,还可在安全容器的配置文件中配置安全容器是共享型的,还是独占型的。相应地,如图1b所示,还可从安全容器的配置文件中,获取安全容器是否为共享型安全容器。若是,则采用本申请实施例提供的容器创建方法创建安全容器,即图1b示出的根据资源请求量及资源限制量,确定安全容器的虚拟处理器核数,即第一数量。具体可采用前述公式(1)来确定。进一步,还可采用前述公式(3)对安全容器的虚拟处理器核数进行扩展,以得到虚拟处理器核的目标数量。还可采用前述公式(2)确定安全容器的限制使用时长(如cpu.quota)。进一步,可根据目标数量的虚拟处理器核,创建安全容器;并设置安全容器的限制使用时长(如cpu.quota)。相应地,若安全容器为独占型安全容器,则可采用前述示出的安全容器创建方法创建安全容器。具体地,可采用前述公式(1)确定安全容器的虚拟处理器核数,即第一数量;并采用前述公式(2)确定安全容器的限制使用时长(如cpu.quota)。进一步,可根据第一数量的虚拟处理器核,创建安全容器;并设置安全容器的限制使用时长(如cpu.quota)。

[0067] 对于创建完成的共享型安全容器,其具有目标数量的虚拟处理器核,目标数量大于根据资源请求量和资源限制量确定出的虚拟处理器核数,为在安全容器的虚拟处理器的负载满足高负载条件时使用弹性处理器带宽控制提供了前提条件。

[0068] 在一些实施例中,可默认安全容器支持弹性处理器带宽控制技术。在另一些实施例中,可在创建出安全容器之后,在安全容器中使能弹性处理器带宽控制技术。由于安全容

器引入了虚拟化层,安全容器与宿主机内核隔离。因此,需要在安全容器以及宿主机这两个层面同步使能弹性处理器带宽控制技术,这样弹性处理器带宽控制技术才能在安全容器中生效。

[0069] 具体地,可在安全容器的配置文件(如yaml文件)中添加特定的命令行(cmdline)参数,来指示客户端操作系统(Guest OS)使能弹性处理器带宽控制技术。其中,客户端操作系统是指安全容器所在的虚拟机实例的操作系统。命令行(cmdline)参数是指在启动时传递给内核的命令行参数,是启动程序或内核时传递的配置选项。在启动安全容器时,可以通过命令行参数来启用弹性带宽控制技术。具体地,可在安全容器的配置文件中添加用于指示客户端操作系统启动弹性处理器带宽控制功能的命令行,如“elastic_cpu=on”。在创建安全容器过程中,通过容器管理工具(如KVM)将该命令行传递至客户端操作系统。客户端操作系统可基于该命令行,启动弹性处理器带宽控制功能,实现安全容器层面的处理器弹性控制功能使能。

[0070] 在宿主机层面通过安全容器的虚拟处理器核扩展机制实现。具体地,在创建安全容器过程中,宿主机的操作系统可通过命令行启动弹性处理器带宽控制功能。

[0071] 在安全容器及宿主机层面均使能弹性处理器带宽控制功能之后,便可通过扩展出的目标数量的虚拟处理器核实现弹性处理器带宽控制。下面对本申请实施例提供的处理器时间调度方法,来实现弹性处理器带宽控制的过程进行具体说明。

[0072] 图2为本申请实施例提供的处理器时间调度方法的流程示意图。如图2所示,该方法主要包括以下步骤:

[0073] 201、在安全容器的虚拟处理器的负载满足高负载条件的情况下,获取安全容器的虚拟处理器在历史调度周期累积的未消耗的第一时长。

[0074] 其中,虚拟处理器具有目标数量的虚拟处理器核;目标数量是在第一数量的基础上扩展得到的;第一数量是根据安全容器的资源请求量和资源限制量确定的。关于第一数量和目标数量的确定方式可参见前述实施例的相关内容,在此不再赘述。

[0075] 202、通过目标数量的虚拟处理器核消耗第一时长,来使用物理处理器执行任务。

[0076] 在本实施例中,针对任一调度周期X,可调用目标数量的虚拟处理器核执行该调度周期X的任务;并统计目标数量的虚拟处理器核完成该调度周期X的任务所消耗的总时间T_x。若目标数量的虚拟处理器核完成该调度周期X的任务所消耗的总时间,小于限制使用时长,则在已累积的未消耗的时间的基础上,增加目标数量的虚拟处理器核在调度周期X未消耗的时间,得到调度周期X对应的未消耗的累积时长。

[0077] 其中,限制使用时长是指安全容器可使用物理处理器的时间上限,等于第一数量乘以调用周期,即:

[0078] cpu.quota = nr_vcpu * cpu.period(4)。

[0079] 其中,cpu.quota是指安全容器可使用的处理器时间上限,即限制使用时长。

[0080] 目标数量的虚拟处理器核在调度周期X未消耗的时间t_x,等于限制使用时长减去目标数量的虚拟处理器核完成该调度周期X的任务所消耗的总时间T_x。即t_x=cpu.quota-T_x。

相应地,调度周期X对应的未消耗的累积时长T_{Xa}可表示为:$T_{Xa}=\sum_{i=1}^{X}t_i$。i表示第i个调度周期。t_i表示目标数量的虚拟处理器核在第i个调度周期X未消耗的时间。

[0081] 相应地,若目标数量的虚拟处理器核完成该调度周期X的任务所消耗的总时间,大于或等于限制使用时长,则确定安全容器的虚拟处理器的负载满足设定的高负载条件,且在调度周期X的下一调度周期(X+1)使用本申请实施例提供的处理器时间调度方法。

[0082] 针对当前调度周期k的前一调度周期(k-1),若目标数量的虚拟处理器核在前一调度周期(k-1)完成任务所消耗的总时间达到安全容器对应的限制使用时长,则确定安全容器的虚拟处理器的负载满足设定的高负载条件。相应地,在该实施方式中,设定的高负载条件实现为:目标数量的虚拟处理器核在前一调度周期(k-1)完成任务所消耗的总时间达到安全容器对应的限制使用时长。

[0083] 在另一些实施例中,还可监测安全容器的处理器利用率。其中,处理器利用率可反映处理器负载水位。若安全容器的处理器利用率大于或等于设定的利用率阈值,则确定安全容器的虚拟处理器的负载满足设定的高负载条件。相应地,在该实施方式中,设定的高负载条件实现为:安全容器的处理器利用率大于或等于设定的利用率阈值。

[0084] 进一步,可在当前调度周期k使用本申请实施例提供的处理器时间调度方法。具体地,可获取安全容器的虚拟处理器在历史调度周期累积的未消耗的时长。在本申请实施例中,将当前调度周期之前的调度周期,称为历史调度周期。其中,若当前调度周期k为检测到安全容器的虚拟处理器的负载满足设定的高负载条件之后的第一个调度周期,则安全容器的虚拟处理器在历史调度周期累积的未消耗的时长可表示为:$T_{(k-1)a}=\sum_{i=1}^{k-1}t_i$,即安全容器的虚拟处理器在历史调度周期累积的未消耗的时长,等于第(k-1)个调度周期对应的未消耗的累积时长。

[0085] 若当前调度周期k为检测到安全容器的虚拟处理器的负载满足设定的高负载条件之后的其它调度周期,则安全容器的虚拟处理器在历史调度周期累积的未消耗的时长,等于检测到安全容器的虚拟处理器的负载满足设定的高负载条件的调度周期对应的未消耗的累积时长,减去检测到安全容器的虚拟处理器的负载满足设定的高负载条件之后的调度周期已消耗的超出该安全容器对应的限制使用时长(如cpu.quota)的时长。限制使用时长(如cpu.quota)是指安全容器使用物理处理器的时间上限。

[0086] 例如,假设检测到安全容器的虚拟处理器的负载满足设定的高负载条件的调度周期为第(k-j)个调度周期,第(k-j)个调度周期对应的未消耗的累积时长可表示为$T_{(k-j)a}$,则安全容器的虚拟处理器在历史调度周期累积的未消耗的时长,等于

$T_{(k-j)a}-\sum_{i=(k-j+1)}^{k-1}T_{i2}$。其中，$T_{i2}$表示第i个调度周期消耗的超出该安全容器对应的限制使用时长(如cpu.quota)的时长。

[0087] 进一步，可通过目标数量的虚拟处理器核，消耗虚拟处理器在历史调度周期累积的未消耗的时长，来使用物理处理器执行任务。

[0088] 在本实施例中，通过对传统安全容器分配到的逻辑核(即虚拟处理器核)进行扩展，打破了传统安全容器中限制使用时长(Quota)与虚拟处理器核需要一比一的限制，安全容器可在高负载时，利用扩展出的虚拟处理器核即扩展逻辑核，消耗历史调度周期累积的未消耗的时长，实现弹性处理器带宽控制，有助于满足突发型负载需求，提高安全容器的服务质量。

[0089] 具体地，针对当前调度周期k，可从虚拟处理器在历史调度周期累积的未消耗的时长中，获取虚拟处理器在当前调度周期k可超限使用物理处理器的时长。在本申请实施例中，为了便于描述和区分，将安全容器的虚拟处理器在历史调度周期累积的未消耗的时长，定义为第一时长；并将虚拟处理器在当前调度周期k可超限使用物理处理器的时长，定义为第二时长T_{k2}。

[0090] 在一些实施例中，可根据扩展倍数及安全容器对应的限制使用时长，确定第二时长T_{k2}。其中，扩展倍数等于目标数量除以前述第一数量。具体地，可计算扩展倍数与限制使用时长(Quota)的乘积，作为安全容器的虚拟处理器在当前调度周期k可使用物理处理器的总时长；进一步，可计算安全容器的虚拟处理器在当前调度周期k可使用物理处理器的总时长，与限制使用时长之间的差值，作为虚拟处理器在当前调度周期k可超限使用物理处理器的时长，即第二时长T_{k2}。由于安全容器的虚拟处理器核的数量增加了扩展倍数倍，因此，将安全容器的虚拟处理器在当前调度周期k可使用物理处理器的总时长，相较于限制使用时长(Quota)也扩大扩展倍数倍，可使多扩展出的的虚拟处理器有足够的物理处理器时间来执行任务，从而提高整体性能。

[0091] 进一步，可从安全容器的虚拟处理器在历史调度周期累积的未消耗的第一时长中，获取第二时长T_{k2}。之后，可在安全容器对应的限制使用时长(如cpu.quota)的基础上增加第二时长T_{k2}，以得到目标时长。其中目标时长=cpu.quota+ T_{k2}。

[0092] 进一步，可为目标数量的虚拟处理器核共分配目标时长的时间，以供目标数量的虚拟处理器核在分配到的时间内使用物理处理器执行当前调度周期k的任务。

[0093] 具体地，可以以目标数量的虚拟处理器核分配到的时间之和等于目标时长为约束条件，根据各虚拟处理器核在当前调度周期执行的任务的资源需求，确定目标数量的虚拟处理器核各自对应的待分配时长。

[0094] 可选地，针对每个虚拟处理器核可在当前调度周期k内，收集该虚拟处理器核执行任务所需的处理器时间。资源需求可以用处理器时间百分比、处理器时间绝对值或其他度量单位表示。进一步，可根据目标数量的虚拟处理器核执行任务所需的处理器时间，计算目标数量的虚拟处理器核的权重。针对任一虚拟处理器核其权重可等于该虚拟处理器核执行任务所需的处理器时间，占目标数量的虚拟处理器核执行任务所需的处理器时间之和的百分比。进一步，可根据目标数量的虚拟处理器核的权重，确定目标数量的虚拟处理器核各自

对应的待分配时长。任一虚拟处理器核对应的待分配时长,等于该虚拟处理器核的权重乘以目标时长。

[0095] 进一步,可为各虚拟处理器核分配对应的待分配时长的时间,来为目标数量的虚拟处理器核共分配目标时长的时间,这样,各虚拟处理器核可在各自分配到的时间内执行当前调度周期k的任务,实现安全容器的弹性处理器带宽控制,有助于满足突发型负载需求,提高安全容器的服务质量。

[0096] 在为目标数量的虚拟处理器核共分配目标时长的时间之后,还可从存储的第一时长中,减去第二时长,作为虚拟处理器在历史调度周期累积的未消耗的新的时长,以供下一调度周期基于新的时长进行处理器时间调度,直至累积的未消耗的时长消耗完毕。在累积的未消耗的时长消耗完毕之后,可继续以目标数量的虚拟处理器核共分配到的总时间等于限制使用时长为约束条件,为目标数量的虚拟处理器核共分配限制使用时长的时间。这样,各虚拟处理器核可在各自分配到的时间内执行相应调度周期的任务。

[0097] 在一些实施例中,当前调度周期k获取到的安全容器的虚拟处理器在历史调度周期累积的未消耗的第一时长,可能小于前述确定出的第二时长T_{k2},则可在安全容器对应的限制使用时长(cpu.quota)的基础上增加第一时长,作为目标数量的虚拟处理器核在当前调度周期共分配到的时长(定义为第四时长)。进一步,可为目标数量的虚拟处理器核共分配第四时长的时间。这样,目标数量的虚拟处理器核在分配到的时间内使用物理处理器执行当前调度周期k的任务,实现安全容器的弹性处理器带宽控制,有助于满足突发型负载需求,提高安全容器的服务质量。

[0098] 其中,关于为目标数量的虚拟处理器核共分配第四时长的时间的具体实施方式,可参见前述为目标数量的虚拟处理器核共分配目标时长的时间的相关内容,在此不再赘述。

[0099] 由于弹性处理器带宽控制技术允许将每个调度周期内未用完的处理器时间累积起来,在处理器高负载期间使用,从而在高负载期间突破了安全容器的处理器配额(cpu.quota)的限制,提高安全容器的服务质量。但这可能会加剧同一宿主机上各容器对处理器资源的争抢,进而影响安全容器的邻居容器的服务质量。其中,安全容器的邻居容器是指与安全容器部署于同一宿主机的其它容器,且与安全容器共享宿主机的物理处理器。因此,可优先保证邻居容器的处理器配额(cpu.quota)得到满足,在此基础上再满足安全容器突破处理器配额(cpu.quota)的需求。基于此,在本申请一些实施例中,如图3所示,引入安全容器的调度权重动态调节机制。

[0100] 具体地,可预先配置宿主机上部署的各容器(包括安全容器和该宿主机上部署的其它容器)的调度权重。其中,预先配置的各容器的调度权重可以相同,也可不同。在一些实施例中,可预先配置的各容器的调度权重相同,等于1/N。N表示该宿主机上部署的容器的总数量,包括安全容器和该宿主机上部署的其它容器。在另一些实施例中,可根据宿主机上部署的各容器的限制使用时长,配置各容器的调度权重。具体地,针对任一容器A,可计算容器A的限制使用时长与各容器的限制使用时长之和的比值,作为容器A的调度权重。或者,可将容器A的限制使用时长作为该容器A的调度权重。前述配置的容器的调度权重为容器的初始调度权重。

[0101] 在安全容器的虚拟处理器负载满足设定的高负载条件的情况下,可降低安全容器

的调度权重,以得到安全容器的目标调度权重。其中,目标调度权重小于安全容器的初始调度权重,且目标调度权重小于该宿主机上部署的其它容器的调度权重。

[0102] 其中,目标调度权重可为预先设置的。目标调度权重小于该宿主机上部署的其它容器的调度权重。这样,可在安全容器的虚拟处理器负载满足设定的高负载条件的情况下,将安全容器的调度权重设置为预先设置的目标调度权重。目标调度权重小于安全容器的初始调度权重。

[0103] 或者,可在安全容器的虚拟处理器负载满足设定的高负载条件的情况下,将安全容器的调度权重降低设定的梯度,得到安全容器的目标调度权重。即在安全容器的虚拟处理器负载满足设定的高负载条件的情况下,将安全容器的当前调度权重减去设定的梯度,得到安全容器的目标调度权重。假设设定梯度为0.05,则将安全容器的当前调度权重减去0.5,得到安全容器的目标调度权重。

[0104] 进一步,可根据安全容器的目标调度权重及该宿主机上部署的其它容器的调度权重,确定安全容器在当前调度周期k使用物理处理器的时长(定义为第三时长)。具体地,可计算目标调度权重与宿主机上部署的各容器的调度权重之和的比值P之后,计算比值P与调度周期(Period)的乘积,得到安全容器在当前调度周期k使用物理处理器的第三时长。

[0105] 进一步,若第三时长小于或等于安全容器对应的限制使用时长(如cpu.quota),则可确定安全容器在当前调度周期k可使用物理处理器的总时长为限制使用时长,则可为目标数量的虚拟处理器核共分配限制使用时长的时间。这样,安全容器的目标数量的虚拟处理器核可在限制使用时长的时间内执行当前调度周期k的任务。

[0106] 相应地,若第三时长大于安全容器对应的限制使用时长(如cpu.quota),则可根据第三时长与安全容器对应的限制使用时长(如cpu.quota)之间的差值,确定安全容器在当前调度周期k可超限使用物理处理器的第二时长。这样,一方面可兼顾安全容器的动态调度权重的调整,优先保证满足其它容器的处理器配额,减少安全容器与其它容器之间的处理器资源冲突,进而降低弹性处理器带宽控制技术对邻居容器的干扰。另一方面,还可继续在安全容器中使用弹性处理器带宽控制技术,突破安全容器对应的限制使用时长(即处理器配额)的限制,有助于提高安全容器的服务质量。

[0107] 在一些实施例中,可将第三时长与安全容器对应的限制使用时长(如cpu.quota)之间的差值,作为安全容器在当前调度周期k可超限使用物理处理器的第二时长。

[0108] 在另一些实施例中,可计算虚拟处理器核的扩展倍数及安全容器对应的限制使用时长的乘积,作为安全容器在当前调度周期k可使用物理处理器的总时长。其中,扩展倍数等于目标数量除以第一数量。之后,可计算安全容器在当前调度周期k可使用物理处理器的总时长与安全容器对应的限制使用时长之间的差值。为了便于描述和区分,将安全容器在当前调度周期k可使用物理处理器的总时长与安全容器对应的限制使用时长(Quota)之间的差值,定义为第一差值,并将第三时长与安全容器对应的限制使用时长(如cpu.quota)之间的差值,定义为第二差值。进一步,可从第一差值和第二差值中,选择最小值,作为安全容器在当前调度周期k可超限使用物理处理器的第二时长。从安全容器在当前调度周期k可使用物理处理器的总时长与安全容器对应的限制使用时长(Quota)之间的第一差值,与第三时长与限制使用时长(Quota)之间的第二差值中,选择较小的作为第二时长,可平衡根据安全容器的动态调度权重确定出的安全容器可超限使用物理处理器的时长,与前述根据扩展

倍数及限制使用时长(Quota)的乘积,确定出的安全容器可超限使用物理处理器的时长不同,选择二者中的最小值可使安全容器超限使用物理处理器的时长仍然受到限制,从而不会过度占用资源。

[0109] 在确定出安全容器在当前调度周期k可超限使用物理处理器的第二时长之后,可从第一时长中获取第二时长,并在安全容器对应的限制使用时长的基础上增加第二时长,以得到目标时长。进一步,可为目标数量的虚拟处理器核共分配目标时长的时间,这样,目标数量的虚拟处理器核在分配到的时间内使用物理处理器执行当前调度周期k的任务,实现安全容器的弹性处理器带宽控制,使得安全容器突破其处理器配额的限制,有助于提高安全容器在处理器负载突发情况下的服务质量。

[0110] 在实际应用中,在一些情况下安全容器需要获取处理器信息进行一些操作。例如,安全容器需要获取虚拟处理器核数进行资源初始化。在一些实施例中,.当容器内的进程启动时,可读取内核接口或系统调度接口进行资源初始化。其中,内核接口可为/proc/cpuinfo接口或/proc/stat接口。系统调用接口可为/sys/devices/system/cpu/online接口、/sys/devices/system/cpu/offline接口、/sys/devices/system/cpu/possible接口或/sys/devices/system/cpu/present等。

[0111] 其中,/proc/cpuinfo是操作系统(如Linux系统)中 /proc 文件系统的一部分,它提供了一种机制来访问内核关于处理器的信息。通过读取/proc/cpuinfo 文件,可以获取有关系统中安装的处理器的详细信息。因此,容器内的应用程序可读取 /proc/cpuinfo内核接口获取处理器信息,以便根据可用的虚拟处理器核数(如CPU核数)初始化线程池。/proc/stat包含了关于系统中各种统计信息的数据,其中包括处理器(如CPU)的统计信息。

[0112] /sys/devices/system/cpu/online用于设置或查看当前系统中哪些虚拟处理器核(如CPU核)是在线的。/sys/devices/system/cpu/offline用于设置或查看当前系统中哪些虚拟处理器核(如CPU核)是离线的。/sys/devices/system/cpu/possible显示系统中所有可能使用的虚拟处理器核(如CPU核)的范围。/sys/devices/system/cpu/present显示系统中实际存在的虚拟处理器核(如CPU核)的范围。

[0113] 由于本申请实施例中对安全容器的虚拟处理器核数进行了扩展,因此,在向安全容器提供处理器信息时,需要隐藏多扩展出的(目标数量-第一数量)个虚拟处理器核,这主要是因为安全容器内位于用户态的应用程序感知到的虚拟处理器核数与安全容器的处理器配额(cpu.quota)之间保持一致,从而避免资源管理上的混淆和不一致性。用户态的应用程序会根据可见的虚拟处理器核数来进行初始化和资源分配。如果用户态的应用程序感知到的虚拟处理器核数与安全容器的处理器配额(cpu.quota)不一致,可能会导致资源管理混乱。通过隐藏额外的扩展出的虚拟处理器核数,可以确保容器视角可见的虚拟处理器核数与安全容器的处理器配额(cpu.quota)是一对一的关系,从而简化资源管理。

[0114] 基于此,在本申请一些实施例中,可响应于安全容器对处理器信息的访问操作,向安全容器提供第一数量的虚拟处理器核的资源视图。安全容器对处理器信息的访问操作可实现为对前述内核接口或系统调用接口的读取操作。

[0115] 如图3所示,为方便描述和区分,在图3中vCPU表示虚拟处理器核,将安全容器视角可见的vCPU记做gvCPU,将安全容器实际的vCPU记做hvCPU。图3中容器创建请求的资源请求量和资源限制量,确定出的虚拟处理器核数(vCPU数量)为4个,即第一数量等于4。扩展倍数

为2,即实际创建出的安全容器的虚拟处理器核数(目标数量)为8个,如图3中的hvCPU0-7。由于多扩展出的虚拟处理器核多用于处理安全容器内的处理器负载突发应用,因此多扩展出的虚拟处理器核可称为突发型虚拟处理器核,如图3中的突发型vCPU。其中,图3中“突发型vCPU创建机制”是指前述容器创建方法提供的安全容器创建过程。在图3中,安全容器内应用程序视角可见的虚拟处理器核数为4个,即图3中的gvCPU0-3。

[0116] 为了实现扩展出的虚拟处理器核的隐藏机制的,在本申请一些实施例中,如图3所示,将安全容器内实际的虚拟处理器核的数量(即目标数量)缩放扩展倍数,得到安全容器视角可见的虚拟处理器核的数量(即第一数量)。为了方便描述,将扩展倍数记为f;安全容器内实际的虚拟处理器核的数量(即目标数量)记为s;安全容器视角内可见虚拟处理器核的数量(即第一数量)记为s’,则有s’=s/f。在图3中,s’=4,s=8,f=2。

[0117] 具体地,如图4所示,在虚拟机的操作系统的内核(Guest OS内核)可使用位图标记虚拟处理器核否在线。位图中的每个比特位对应一个虚拟处理器核,1代表虚拟处理器核在线,0代表该必特位对应的虚拟处理器核不在线。或者,0代表虚拟处理器核在线,1代表该必特位对应的虚拟处理器核不在线等。如图4所示,隐藏多扩展出的虚拟处理器核,可实现为保留位图中[0,s’)这些比特位,清除[s’,s)这些比特位,进而使得安全容器视角可见的虚拟处理器核的数量为第一数量。

[0118] 基于此,可响应于安全容器对处理器信息的访问操作,确定目标数量的虚拟处理器核对应的第一位图。第一位图如图4中(a)图所示。第一位图的一个比特位对应一个虚拟处理器核;在图4中,比特位的值为1表示该比特位对应有虚拟处理器核,比特位的值为0表示该比特位未对应有虚拟处理器核。进一步,可根据扩展倍数,将第一位图修改为第二位图;第二位图中目标比特位的数量等于第一数量;目标比特位是指表征对应有虚拟处理器核的比特位。第二位图如图4中(b)图所示,图4中(b)图中值为1的比特位表示目标比特位。进一步,可将第二位图提供给安全容器,这样可向安全容器提供第一数量的虚拟处理器核的资源视图。安全容器感知到第一数量的虚拟处理器核,可根据第一数量的虚拟处理器核进行资源管理等,这样,可隐藏额外的扩展出的虚拟处理器核数,可以确保容器视角可见的虚拟处理器核数与安全容器的处理器配额(cpu.quota)是一对一的关系,避免资源管理上的混淆和不一致性。

[0119] 为了确保不同应用程序之间的隔离性,可以将不同应用程序绑定到不同的虚拟处理器核上,避免资源争抢。当应用程序绑定到一个安全容器视角的虚拟处理器核(即gvCPU)时,该应用程序实际在前述扩展倍数(即f)个hvCPU上云霄,从而充分利用处理器资源。

[0120] Burst vCPU映射机制的设置思路是将f个hvCPU映射为一个gvCPU。当应用或服务绑定到一个gvCPU上时,该应用或服务实际上可在f个hvCPU上运行,从而充分利用CPU资源;并且这一过程对应用或服务透明。映射关系按下面的公式进行计算:

[0121] $$gvCPU(i)=\sum_{j=0}^{f-1} hvCPU(i+j\bullet s') \quad (5)。$$

[0122] 在式(5)中,i表示第i个gvCPU,即容器视角可见的第i个虚拟处理器核。假设hvCPU总数(即目标数量)s为8,扩展倍数为2,则gvCPU0=hvCPU0+hvCPU4。

[0123] 具体地,可使用位图维护应用程序的绑核信息。某一比特位为1表示应用程序可运行在该比特位对应的虚拟处理器核上,某一比特位为0表示应用程序不能运行在该比特位对应的虚拟处理器核上。在应用程序设置绑核信息时,突发型vCPU映射机制可按上述公式中的关系将应用分散绑定到f(即扩展倍数)个hvCPU上;在应用程序读取绑核信息时,突发型vCPU映射机制按上述公式中的关系将f个hvCPU聚合到一个gvCPU上。

[0124] 基于上述突发型vCPU映射机制,可响应于安全容器的处理器绑定操作,获取处理器绑定操作待绑定的目标虚拟处理器核及待绑定的进程。其中,安全容器可读取内核接口或系统调用接口进行处理器绑定操作。内核接口可为/sys/fs/cgroup/cpuset/cpuset.cpus或/sys/fs/cgroup/cpuset/cpuset.effective_cpus等。系统调用接口可为sched_setaffinity或sched_getaffinity等。

[0125] 其中,/sys/fs/cgroup/cpuset/cpuset.cpus用于指定一组处理器核(如CPU核),这些处理器核(如CPU核)可以被资源控制组(Cgroup)中的进程使用。/sys/fs/cgroup/cpuset/cpuset.effective_cpus用于显示当前资源控制组(Cgroup)中进程实际可以使用的处理器核(如CPU核)集合。

[0126] sched_setaffinity是一个系统调用,用于设置进程的处理器(如CPU)的亲和性。这意味着它可以指定一个进程只能在哪些处理器核(如CPU核)上运行。ched_getaffinity是一个系统调用,用于获取进程的当前处理器(如CPU)的亲和性设置。

[0127] 进一步,可根据扩展倍数f,确定目标虚拟处理器核对应的目标比特位在前述第一位图中对应的f(即扩展倍数)个比特位。可按照前述公式(1)确定目标虚拟处理器核对应的目标比特位在前述第一位图中对应的f(即扩展倍数)个比特位。如图5所示,gvCPU0对应的目标比特位在前述第一位图中对应的f个比特位为hvCPU0和hvCPU4对应的比特位。进一步,可将待绑定的进程与确定出的f扩展倍数个比特位对应的虚拟处理器核进行绑定,这样可将待绑定的进程运行在扩展倍数(即f)个虚拟处理器核上,充分利用处理器资源,提供处理器资源利用率。

[0128] 前述实施例提供的多扩展出的虚拟处理器核隐藏与映射机制。该机制隐藏为支持弹性处理器带宽控制技术而多扩展的虚拟处理器核,并对容器视角可见虚拟处理器核与多扩展出的虚拟处理器核进行合理映射,从而实现多扩展的虚拟处理器核对安全容器内的应用程序透明,安全容器内的应用程序无改造成本。

[0129] 值得说明的是,本申请实施例提供的安全容器可运行各种服务,如数据库服务、在线购物服务、视频服务或云计算服务等,但不限于此。本申请发明人将本申请实施例提供的安全容器应用在数据库上,并使用本申请实施例提供的容器创建方法创建数据库对应的安全容器,以及使用本申请实施例提供的处理器时间调度方法对数据库对应的安全容器进行资源调度,并使用事务处理性能委员会基准测试H(Transaction Processing Performance Council Benchmark H,TPC-H)对数据库进行性能评估。其中,TPC-H是一项广泛使用的基准测试,用于评估数据库系统的性能。TPC-H基准测试涵盖了多种复杂的查询模式,旨在模拟真实世界中的数据分析和报告任务。

[0130] TPC-H的主要指标包括:(1)持续时间(Duration)指标,是指完成TPC-H基准测试中所有查询所需的总时间,该指标越低越好;其中,Duration越低,表示系统完成所有查询的速度越快,性能越高;(2)吞吐量(Throughput)指标是指单位时间内系统能够完成的工作

量,通常是每小时完成的查询数量,该指标越高越好;Throughput越高,表示系统在单位时间内能够处理更多的查询,性能越强。

[0131] 测试结果显示,在在数据库配置不变的情况下,使用本申请实施例提供的容器创建方法创建出的数据库对应的安全容器,并使用本申请实施例提供的处理器时间调度方法对数据库对应的安全容器进行资源调度,使得数据库的TPC-H的持续时间(Duration)指标降低32%,吞吐量(Throughput)指标提升45%。

[0132] 需要说明的是,上述实施例所提供方法的各步骤的执行主体均可以是同一设备,或者,该方法也由不同设备作为执行主体。比如,步骤201和202的执行主体可以为设备A;又比如,步骤201的执行主体可以为设备A,步骤202的执行主体可以为设备B;等等。

[0133] 另外,在上述实施例及附图中的描述的一些流程中,包含了按照特定顺序出现的多个操作,但是应该清楚了解,这些操作可以不按照其在本文中出现的顺序来执行或并行执行,操作的序号如201、202等,仅仅是用于区分开各个不同的操作,序号本身不代表任何的执行顺序。另外,这些流程可以包括更多或更少的操作,并且这些操作可以按顺序执行或并行执行。

[0134] 相应地,本申请实施例还提供一种存储有计算机指令的计算机可读存储介质,当计算机指令被一个或多个处理器执行时,致使一个或多个处理器执行上述容器创建方法和/或处理器时间调度方法中的步骤。

[0135] 本申请实施例还一种计算机程序产品,包括计算机程序,当所述计算机程序被一个或多个处理器执行时,致使所述一个或多个处理器执行上述容器创建方法和/或处理器时间调度方法中的步骤。在本申请实施例中,不限定计算机程序产品的具体实现形态。在一些实施例种,计算机程序产品可实现为应用程序(Application,APP)、小程序、电脑侧客户端、程序模块、插件、安装包、软件开发工具包(Software Development Kit,SDK)、光盘的镜像文件(如ISO文件)、插件或软件即服务((Software as a Service,SaaS)形态的软件等,但不限于此。

[0136] 图6为本申请实施例提供的电子设备的结构示意图。如图6所示,该电子设备包括:存储器60a和处理器60b。其中,所述存储器60a,用于存储计算机程序。

[0137] 处理器60b耦合至存储器60a,用于执行计算机程序以用于执行前述各实施例提供的容器创建方法和/或处理器时间调度方法中的步骤。关于各步骤的具体实施方式可参见前述实施例的相关描述,在此不再赘述。

[0138] 在一些可选实施方式中,如图6所示,该电子设备还可以包括:通信组件60c、电源组件60d、显示组件60e及音频组件60f等可选组件。图6中仅示意性给出部分组件,并不意味着电子设备必须包含图6所示全部组件,也不意味着电子设备只能包括图6所示组件。

[0139] 另外,图6中虚线框内的组件为可选组件,而非必选组件,具体可视电子设备的产品形态而定。本实施例的电子设备可以实现为台式电脑、笔记本电脑、手机或物联网设备等终端设备;也可以是传统服务器、云服务器或服务器集群等各种服务器设备。

[0140] 在本申请实施例中,存储器用于存储计算机程序,并可被配置为存储其它各种数据以支持在其所在设备上的操作。其中,处理器可执行存储器中存储的计算机程序,以实现相应控制逻辑。存储器可以由任何类型的易失性或非易失性存储设备或者它们的组合实现,如静态随机存取存储器(Static Random-Access Memory,SRAM),电可擦除可编程只读

存储器(Electrically Erasable Programmable Read Only Memory,EEPROM),可擦除可编程只读存储器(Electrical Programmable Read Only Memory,EPROM),可编程只读存储器(Programmable Read Only Memory,PROM),只读存储器(Read Only Memory,ROM),磁存储器,快闪存储器,磁盘或光盘。

[0141] 在本申请实施例中,处理器可以为任意可执行上述方法逻辑的硬件处理设备。可选地,处理器可以为中央处理器(Central Processing Unit,CPU)、图形处理器(Graphics Processing Unit,GPU)或微控制单元(Microcontroller Unit,MCU);也可以为现场可编程门阵列(Field-Programmable Gate Array,FPGA)、可编程阵列逻辑器件(Programmable Array Logic,PAL)、通用阵列逻辑器件(General Array Logic,GAL)、复杂可编程逻辑器件(Complex Programmable Logic Device,CPLD)等可编程器件;或者为先进精简指令集(Reduced Instruction Set Compute,RISC)处理器(Advanced RISC Machines,ARM)或系统芯片(System on Chip,SoC)等等,但不限于此。

[0142] 在本申请实施例中,通信组件被配置为便于其所在设备和其他设备之间有线或无线方式的通信。通信组件所在设备可以接入基于通信标准的无线网络,如无线保真(Wireless Fidelity,WiFi),2G或3G,4G,5G或它们的组合。在一个示例性实施例中,通信组件经由广播信道接收来自外部广播管理系统的广播信号或广播相关信息。在一个示例性实施例中,通信组件还可基于近场通信(Near Field Communication,NFC)技术、射频识别(Radio Frequency Identification,RFID)技术、红外数据协会(Infrared Data Association,IrDA)技术、超宽带(Ultra Wide Band,UWB)技术、蓝牙(Bluetooth,BT)技术或其他技术来实现。

[0143] 在本申请实施例中,显示组件可以包括液晶显示器(Liquid Crystal Display,LCD)和触摸面板(Touch Panel,TP)。如果显示组件包括触摸面板,显示组件可以被实现为触摸屏,以接收来自用户的输入信号。触摸面板包括一个或多个触摸传感器以感测触摸、滑动和触摸面板上的手势。触摸传感器可以不仅感测触摸或滑动动作的边界,而且还检测与触摸或滑动操作相关的持续时间和压力。

[0144] 在本申请实施例中,电源组件被配置为其所在设备的各种组件提供电力。电源组件可以包括电源管理系统,一个或多个电源,及其他与为电源组件所在设备生成、管理和分配电力相关联的组件。

[0145] 在本申请实施例中,音频组件可被配置为输出和/或输入音频信号。例如,音频组件包括一个麦克风(Microphone,MIC),当音频组件所在设备处于操作模式,如呼叫模式、记录模式和语音识别模式时,麦克风被配置为接收外部音频信号。所接收的音频信号可以被进一步存储在存储器或经由通信组件发送。在一些实施例中,音频组件还包括一个扬声器,用于输出音频信号。例如,对于具有语言交互功能的设备,可通过音频组件实现与用户的语音交互等。

[0146] 需要说明的是,本文中的“第一”、“第二”等描述,是用于区分不同的消息、设备、模块等,不代表先后顺序,也不限定“第一”和“第二”是不同的类型。

[0147] 本领域内的技术人员应明白,本申请的实施例可提供为方法、系统、或计算机程序产品。因此,本申请可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且,本申请可采用在一个或多个其中包含有计算机可用程序代码的计算机

可用存储介质(包括但不限于磁盘存储器、只读光盘(Compact Disc Read-Only Memory,CD-ROM)、光学存储器等)上实施的计算机程序产品的形式。

[0148] 本申请是参照根据本申请实施例的方法、设备(系统)、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器,使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

[0149] 这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中,使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品,该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

[0150] 这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上,使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理,从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

[0151] 在一个典型的配置中,计算设备包括一个或多个处理器 (CPU等)、输入/输出接口、网络接口和内存。

[0152] 内存可能包括计算机可读介质中的非永久性存储器,随机存取存储器 (Random-Access Memory,RAM) 和/或非易失性内存等形式,如只读存储器 (ROM) 或闪存(flash RAM)。内存是计算机可读介质的示例。

[0153] 计算机的存储介质为可读存储介质,也可称为可读介质。可读存储介质包括永久性和非永久性、可移动和非可移动媒体可以由任何方法或技术来实现信息存储。信息可以是计算机可读指令、数据结构、程序的模块或其他数据。计算机的存储介质的例子包括,但不限于相变内存 (Phase-Change Memory,PRAM)、静态随机存取存储器 (SRAM)、动态随机存取存储器 (Dynamic Random Access Memory,DRAM)、其他类型的随机存取存储器(RAM)、只读存储器 (ROM)、电可擦除可编程只读存储器 (EEPROM)、快闪记忆体或其他内存技术、只读光盘只读存储器 (CD-ROM)、数字多功能光盘 (Digital Video Disc,DVD) 或其他光学存储、磁盒式磁带,磁盘存储或其他磁性存储设备或任何其他非传输介质,可用于存储可以被计算设备访问的信息。按照本文中的界定,计算机可读介质不包括暂存电脑可读媒体 (transitory media),如调制的数据信号和载波。

[0154] 还需要说明的是,术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含,从而使得包括一系列要素的过程、方法、商品或者设备不仅包括那些要素,而且还包括没有明确列出的其他要素,或者是还包括为这种过程、方法、商品或者设备所固有的要素。在没有更多限制的情况下,由语句“包括一个……”限定的要素,并不排除在包括上述要素的过程、方法、商品或者设备中还存在另外的相同要素。

[0155] 以上内容仅为本申请的实施例而已,并不用于限制本申请。对于本领域技术人员来说,本申请可以有各种更改和变化。凡在本申请的精神和原理之内所作的任何修改、等同替换、改进等,均应包含在本申请的权利要求范围之内。

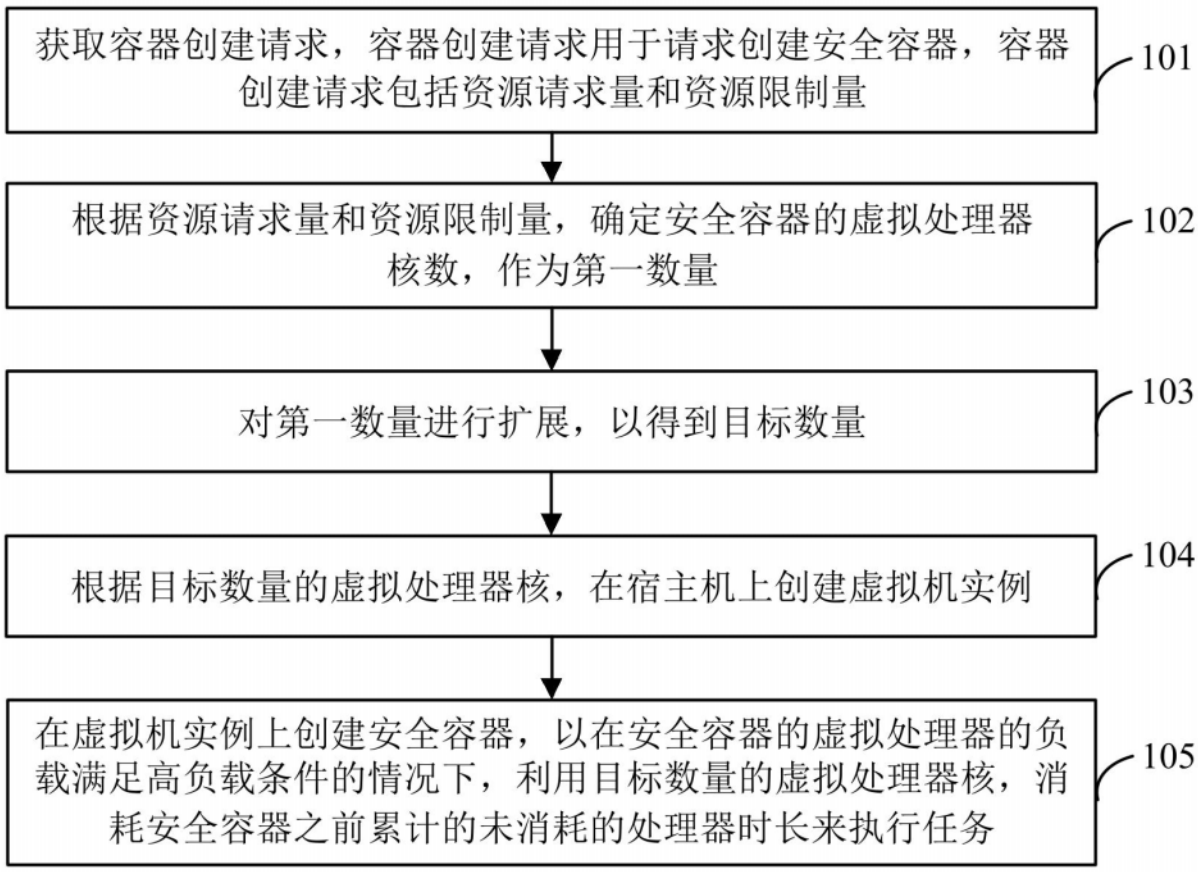

获取容器创建请求，容器创建请求用于请求创建安全容器，容器创建请求包括资源请求量和资源限制量
101
根据资源请求量和资源限制量，确定安全容器的虚拟处理器核数，作为第一数量
102
对第一数量进行扩展，以得到目标数量
103
根据目标数量的虚拟处理器核，在宿主机上创建虚拟机实例
104
在虚拟机实例上创建安全容器，以在安全容器的虚拟处理器的负载满足高负载条件的情况下，利用目标数量的虚拟处理器核，消耗安全容器之前累计的未消耗的处理器时长来执行任务
105

图1a

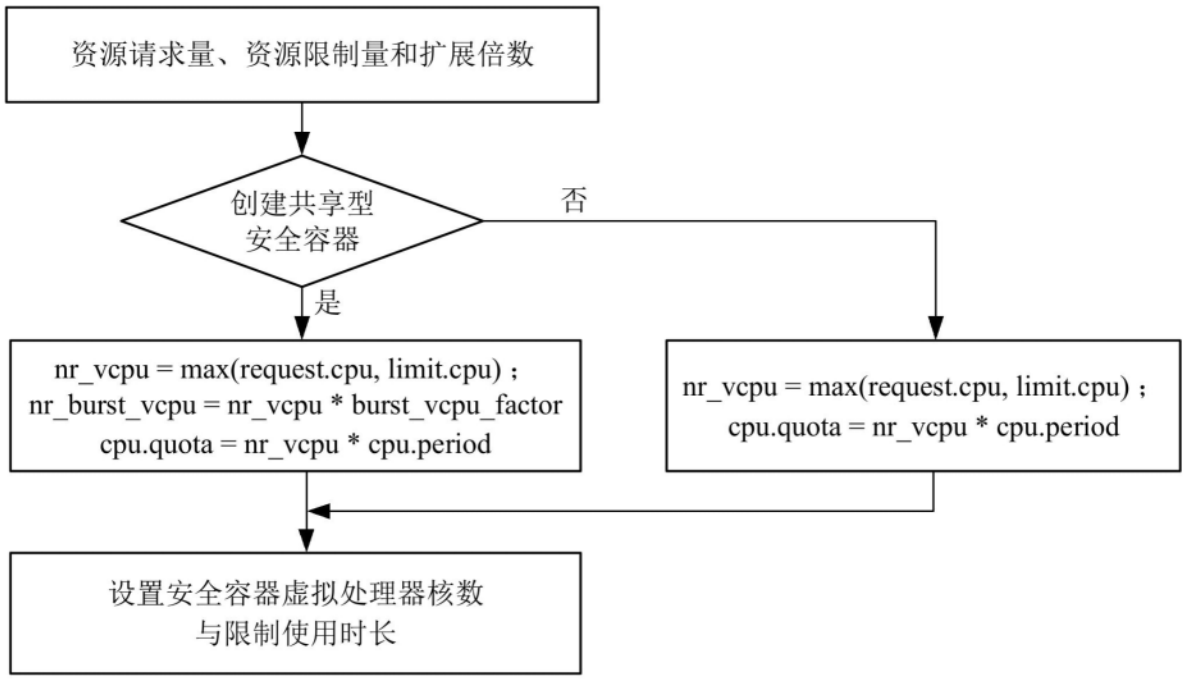

资源请求量、资源限制量和扩展倍数
创建共享型安全容器
否
是
nr_vcpu = max(request.cpu, limit.cpu)；
nr_burst_vcpu = nr_vcpu * burst_vcpu_factor
cpu.quota = nr_vcpu * cpu.period
nr_vcpu = max(request.cpu, limit.cpu)；
cpu.quota = nr_vcpu * cpu.period
设置安全容器虚拟处理器核数与限制使用时长

图1b

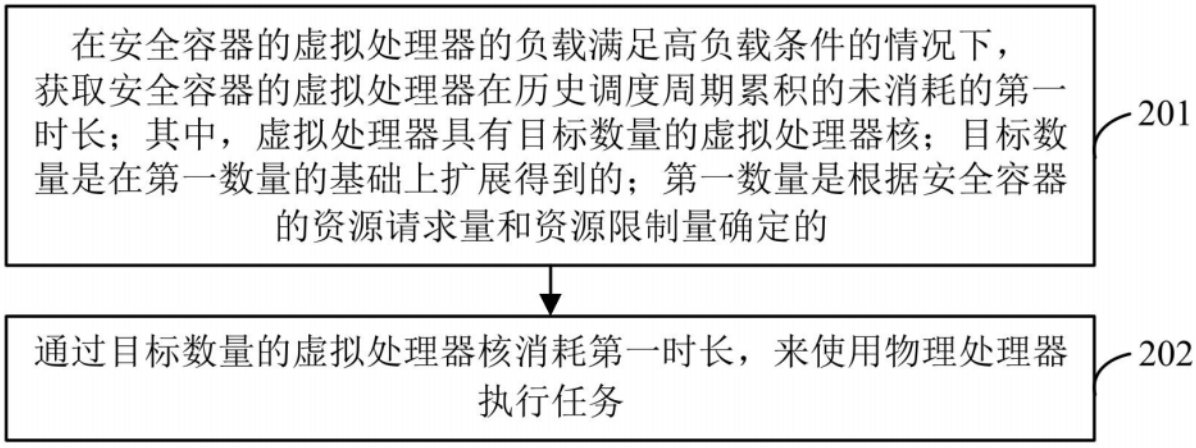

在安全容器的虚拟处理器的负载满足高负载条件的情况下，获取安全容器的虚拟处理器在历史调度周期累积的未消耗的第一时长；其中，虚拟处理器具有目标数量的虚拟处理器核；目标数量是在第一数量的基础上扩展得到的；第一数量是根据安全容器的资源请求量和资源限制量确定的
201
通过目标数量的虚拟处理器核消耗第一时长，来使用物理处理器执行任务
202

图2

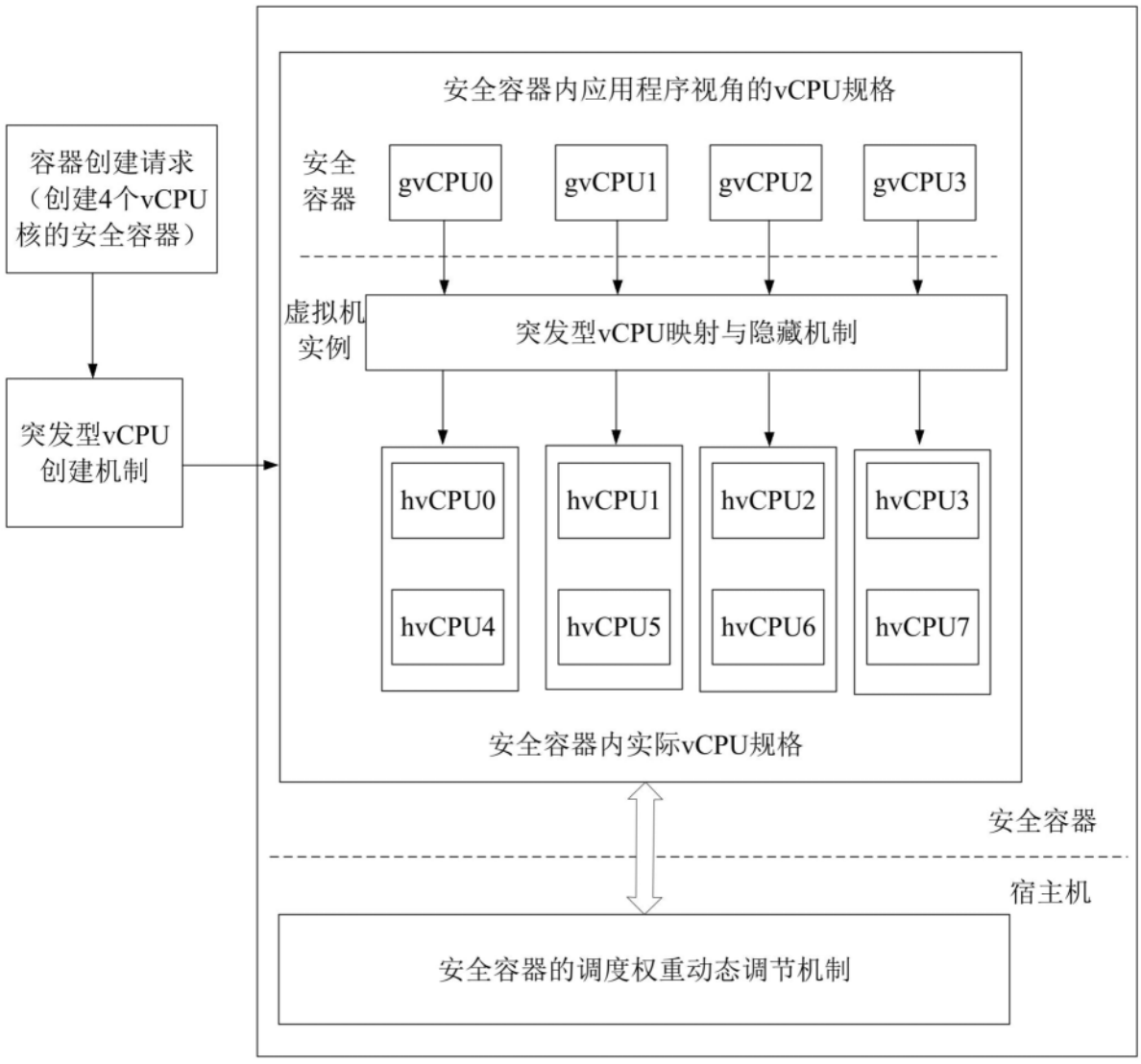

容器创建请求（创建4个vCPU核的安全容器）
突发型vCPU创建机制
安全容器内应用程序视角的vCPU规格
安全容器
gvCPU0
gvCPU1
gvCPU2
gvCPU3
虚拟机实例
突发型vCPU映射与隐藏机制
hvCPU0
hvCPU1
hvCPU2
hvCPU3
hvCPU4
hvCPU5
hvCPU6
hvCPU7
安全容器内实际vCPU规格
安全容器
宿主机
安全容器的调度权重动态调节机制

图3

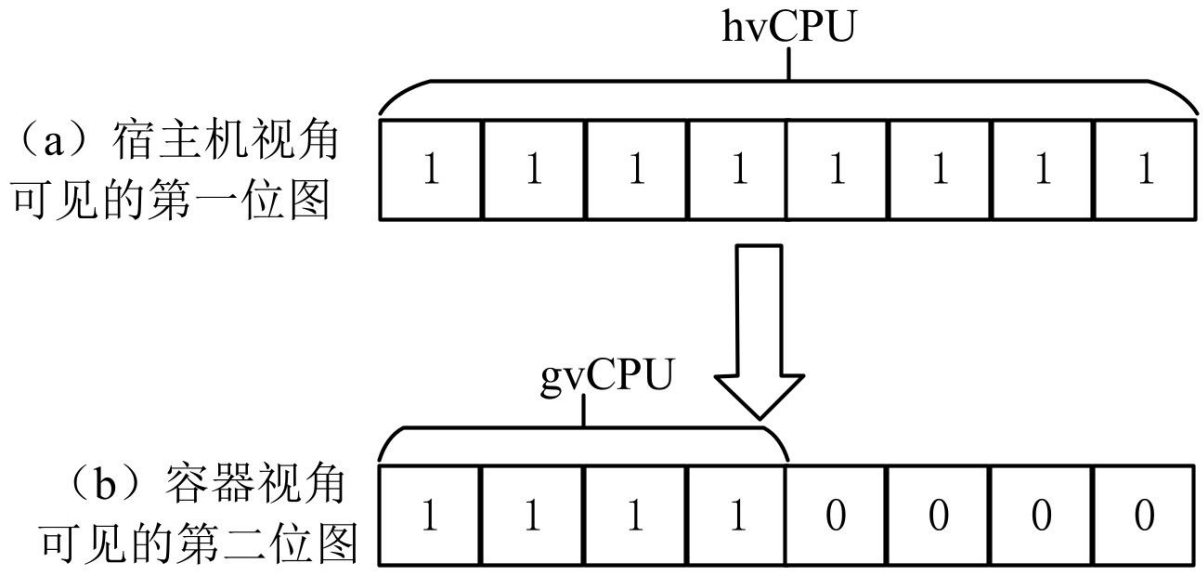

hvCPU
（a）宿主机视角
可见的第一位图
1 1 1 1 1 1 1 1
gvCPU
（b）容器视角
可见的第二位图
1 1 1 1 0 0 0 0

图4

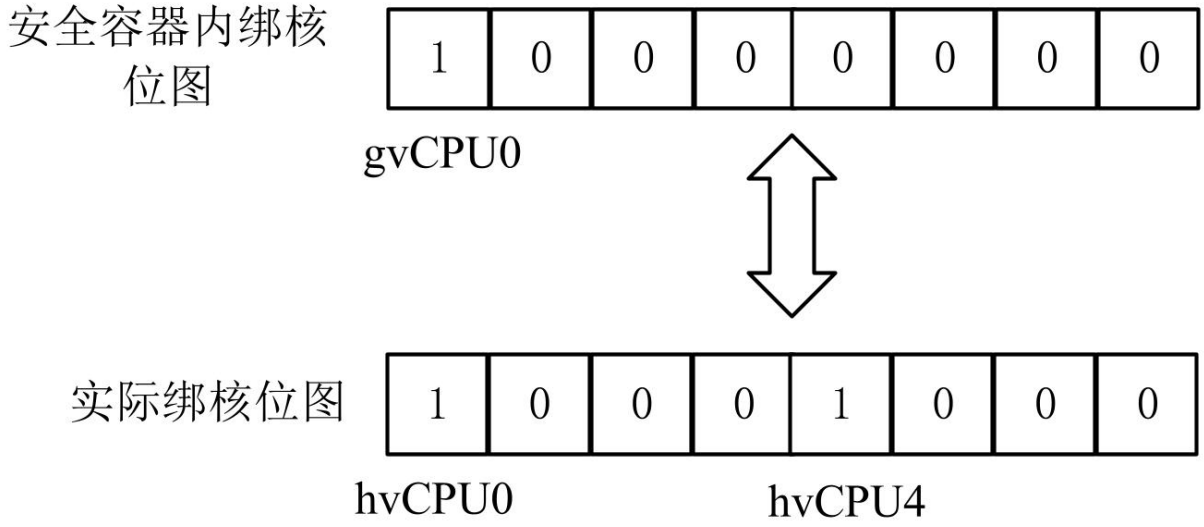

安全容器内绑核
位图
1 0 0 0 0 0 0 0
gvCPU0
实际绑核位图
1 0 0 0 1 0 0 0
hvCPU0
hvCPU4

图5

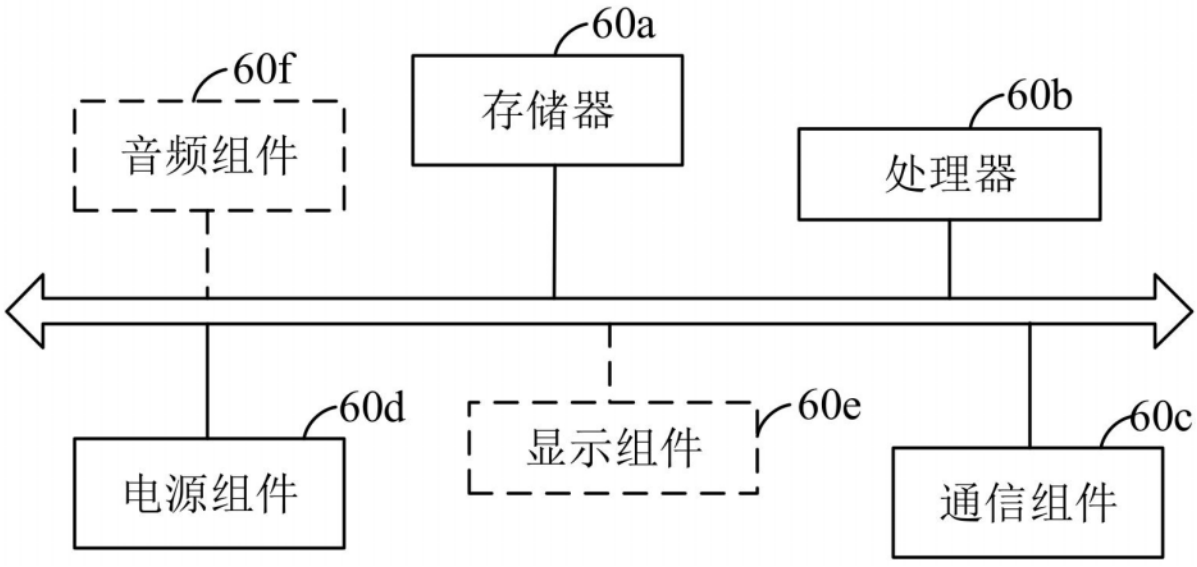


图6