(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 111786904 B
(45)授权公告日 2021.07.06

(21)申请号 202010644567.8
(22)申请日 2020.07.07
(65)同一申请的已公布的文献号
申请公布号 CN 111786904 A
(43)申请公布日 2020.10.16
(73)专利权人 上海道客网络科技有限公司
地址 200433 上海市杨浦区伟德路6号1305-12室
(72)发明人 潘远航 徐俊杰 颜开 熊中祥
(74)专利代理机构 上海市汇业律师事务所 31325
代理人 王函
(51)Int.Cl.
H04L 12/851(2013.01)
H04L 29/08(2006.01)
G06F 9/4401(2018.01)

(56)对比文件
CN 110737503 A,2020.01.31
CN 105897869 A,2016.08.24
CN 110109649 A,2019.08.09
CN 110716758 A,2020.01.21
CN 105245373 A,2016.01.13
US 2015242228 A1,2015.08.27

审查员 李晓利

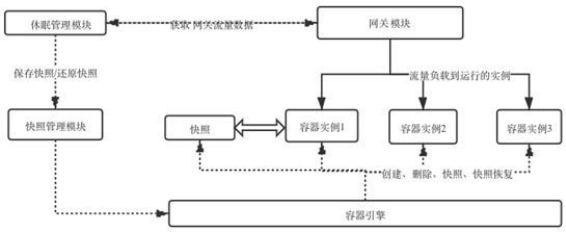
权利要求书2页 说明书9页 附图3页

(54)发明名称

一种实现容器休眠与唤醒的系统及休眠与唤醒方法

(57)摘要

本发明公开了一种实现容器休眠与唤醒的系统及休眠与唤醒方法，系统包含容器引擎、网关模块、快照管理模块、休眠管理模块以及一个或多个应用，所述应用由一个或多个容器组成；休眠方法包含流量进入网关模块后网关模块根据流量请求的信息识别出流量转发的目标应用，网关模块将请求转发至目标应用的容器中；休眠管理模块根据从网关模块获取的访问请求数据与休眠策略结合判断是否有应用需要休眠，并生成传输至快照管理模块的休眠指令；快照管理模块接收休眠指令并依据该休眠指令对相应的容器进行休眠，快照管理模块从容器管理平台查询该容器所在的主机并调用该主机的容器引擎将该容器利用CRIU技术保存成快照；快照管理模块生成关系表。



CN 111786904 B

1.一种实现容器休眠与唤醒的系统,其特征在于,包含容器引擎、网关模块、快照管理模块、休眠管理模块以及一个或多个应用,所述应用由一个或多个容器组成;所述实现容器休眠与唤醒的系统在serverless的架构下运行;

所述网关模块用于接收流量请求并根据所述流量请求生成访问请求数据并对所述应用进行流量的接入、转发以及生成唤醒请求并将所述唤醒请求与所述访问请求数据传输至所述休眠管理模块;

所述休眠管理模块用于与所述网关模块创建连接获取所述访问请求数据,并判断所述应用是否需要进入休眠状态并生成休眠指令,还用于传输所述休眠指令至所述快照管理模块,还用于接收所述唤醒请求并依据所述唤醒请求生成传输至所述快照管理模块的恢复指令,还用于与所述网关模块创建连接同步所述应用的所述休眠指令与所述恢复指令;所述休眠管理模块用于维护应用访问的过期时间表:当应用被访问的时候,所述应用的过期时间会被重置;当所述休眠管理模块发现所述应用过期时,所述休眠管理模块会触发所述快照管理模块休眠所述应用,所述休眠管理模块会通知所述网关模块,对应应用进入休眠状态;

所述快照管理模块用于接收、管理快照,还用于维护所述快照与所述容器的对应关系,还用于接收所述休眠指令并根据所述休眠指令对所述容器进行休眠,还用于接收所述恢复指令对所述容器进行恢复,所述快照管理模块还用于触发所述容器引擎工作;所述快照管理模块接收到所述休眠指令时还用于查询所述容器所在的主机,并调用所述容器所在主机的所述容器引擎将所述容器利用CRIU保存成快照;

所述容器引擎通过CRIU对所述容器进行快照的创建、恢复,并将所述快照传输至所述快照管理模块。

2.如权利要求1所述的一种实现容器休眠与唤醒的系统,其特征在于,所述网关模块内置有流量识别模块;

所述流量识别模块用于识别所述流量请求确定目标应用,所述网关模块将所述流量转发至所述目标应用。

3.如权利要求2所述的一种实现容器休眠与唤醒的系统,其特征在于,所述休眠管理模块内置有过期时间管理模块;

所述过期时间管理模块用于记录所述应用在设定的时间内接收到的请求,并生成所述休眠指令。

4.如权利要求3所述的一种实现容器休眠与唤醒的系统,其特征在于,所述快照管理模块内置有关系表创建模块;

所述关系表创建模块用于维护所述快照与所述容器的对应关系并生成关系表。

5.如权利要求1-4任一项所述的一种实现容器休眠与唤醒的系统,其特征在于,所述休眠管理模块内置有快照恢复模块与过期时间重置模块;

所述快照恢复模块用于接收所述恢复指令时对相应的所述快照中恢复所述容器;

所述过期时间重置模块用于所述容器恢复后对该容器进行过期时间的重置;

所述休眠管理模块还用于所述容器的状态传输至所述网关模块,所述网关模块将接收到的上述请求转发至唤醒后的所述容器中。

6.如权利要求5所述的一种实现容器休眠与唤醒的系统,其特征在于,所述快照恢复模

块根据所述容器的编号或名称确定所述快照后调用所述容器引擎的API进行所述快照的恢复。

7.一种实现容器休眠的方法,其特征在于,包含以下步骤,且在serverless的架构下运行:

步骤A1:流量进入网关模块后网关模块根据流量请求的信息识别出流量转发的目标应用,网关模块将请求转发至目标应用的容器中;

步骤A2:休眠管理模块根据从网关模块获取的访问请求数据与休眠策略结合判断是否有应用需要休眠,并生成传输至快照管理模块的休眠指令,休眠管理模块与网关模块连接进行休眠指令的同步;所述休眠管理模块维护应用访问的过期时间表:当应用被访问的时候,所述应用的过期时间会被重置;当所述休眠管理模块发现所述应用过期时,所述休眠管理模块会触发所述快照管理模块休眠所述应用,所述休眠管理模块会通知所述网关模块,对应应用进入休眠状态;

步骤A3:快照管理模块接收休眠指令并依据该休眠指令对相应的容器进行休眠,同时快照管理模块从容器管理平台查询该容器所在的主机并调用该主机的容器引擎将该容器利用CRIU技术保存成快照;

步骤A4:快照管理模块维护快照与容器之间对应关系的同时生成关系表。

8.如权利要求7所述的一种实现容器休眠的方法,其特征在于,步骤2中判断是否有应用需要休眠的方法为:

获取应用在设定范围内的任何请求并统计个数,当请求个数为0时,则生成休眠指令。

9.一种实现容器唤醒的方法,其特征在于,包含以下步骤,且在serverless的架构下运行:

步骤B1:当网关模块接收到休眠容器的请求时,网关模块保持该请求并生成发送至休眠管理模块唤醒请求;

步骤B2:休眠管理模块接收到唤醒请求后生成传输至快照管理模块的恢复请求;

步骤B3:快照管理模块依据恢复请求对相应的容器进行恢复,并重置容器的过期时间;

步骤B4:快照管理模块根据容器的编号或名称确认快照ID并调用容器引擎的API恢复快照;

步骤B5:容器从快照恢复完毕后,休眠管理模块与网关模块连接同步容器的状态,网关模块将请求转发给容器;

其中,所述休眠管理模块维护应用访问的过期时间表:当应用被访问的时候,所述应用的过期时间会被重置;当所述休眠管理模块发现所述应用过期时,所述休眠管理模块会触发所述快照管理模块休眠所述应用,所述休眠管理模块会通知所述网关模块,对应应用进入休眠状态。

10.一种系统处理装置,其特征在于,包含至少一个处理器与所述至少一个处理器耦合的存储器;

所述存储器存储有可执行指令,其中,所述可执行指令在被所述至少一个处理器执行时使得实现根据权利要求7至9中任一项所述的方法。

11.一种计算机可读存储介质,其特征在于,所述计算机可读存储介质上存储有计算机程序,所述计算机程序被处理器执行时实现如权利要求7至9中任一项所述的方法的步骤。

一种实现容器休眠与唤醒的系统及休眠与唤醒方法

技术领域

[0001] 本发明涉及云原生领域，具体涉及一种实现容器休眠与唤醒的系统及休眠与唤醒方法。

背景技术

[0002] 在云原生领域中，serverless(无服务器)是新兴的，并被广受关注的技术架构技术，它能够降低运营和开发成本，并实现按需扩展等优点，无服务系统 Serverless 也是云计算的方向之一，最常见的应用领域为函数计算又称Faas，全称Function as a service，是Serverless 的子集，也是整个Serverless 的核心；

[0003] Serverless 中文的含义是"无服务器"，是开发者将服务器逻辑运行在无状态的计算容器中，完全由第三方管理的，Faas 具备细粒度调用，实时伸缩，无需关心底层基础设施等特性；

[0004] 基于容器来实现serverless的技术架构中，为达到降低资源开销和自主扩展的目的，serverless架构能在容器内应用的负载(比如API请求次数)达到一定程度的阈值后，自动将应用进行横向扩展(增加容器实例数)，提高其整体承载能力；同理，当应用低负载或者零负载时，可以将应用进行缩容(减少容器实例数)，甚至降低到0个实例(类似于“休眠”)；

[0005] 但是业界存在一个难点，当应用的实例数降低到0之后，当有新的请求，需要“唤醒”该应用时，往往应用重新启动时间较长，无法快速响应请求，造成应用性能和用户体验的下降；

[0006] Serverless(无服务器架构)是指服务端逻辑由开发者实现，运行在无状态的计算容器中，由事件触发，完全被第三方管理，其业务层面的状态则存储在数据库或其他介质中，其优点在于降低运营成本和开发成本，具有自主扩展能力，更简单的管理，其中目前落地的技术中，主要是以AWS的Lambda为代表的FaaS(Function as Service)实现，以及开源的Knative(基于容器和Kubernetes)的实现；

[0007] 基于容器的实现中，为达到降低资源开销和自主扩展，serverless架构能在容器内应用的负荷达到一定程度的阈值后，自动将应用进行横向扩展，增加容器实例数，提高其整体承载能力，同理，当应用低负载或者零负载时，可以将应用进行缩容，减少容器实例数，甚至降低到0个实例，达到类似“休眠”的效果；

[0008] 如图4所示，假定应用是一个网页服务器，当访问网站的请求数增大到300个/秒，serverless框架会将1个应用实例变为3个；

[0009] 如图5所示，当请求数减少到100个/秒，serverless框架又会将3个应用实例减少为1个，以节省资源，让给其他的应用；

[0010] 如图6所示，当请求数降低为0时，过了一定时间，serverless框架会将唯一的实例停止，该应用的实例数变为0，进入“休眠”状态；

[0011] 如图7所示，但是当请求数又开始增加，这时serverless框架会“唤醒”应用，即创建出一个该应用的实例，并将请求导向它；

[0012] 对于无服务框架来说，基本唤醒的过程如图8所示，拉取镜像根据镜像大小和网速需要一定时间，配置网络准备启动服务，通常启动服务到能够提供服务还有一些初始化的过程，上面的过程在一些轻量级应用程序场景中，比如Golang、Python 程序，通常只需要数秒钟就能完成，但是在一些重框架应用程序场景就会遇到极大挑战，例如目前大部分的微服务场景都是基于Java语言编写，应用的启动和预热非常耗时(例如Java的Spring框架会去扫描所有的类库，逐个加载，冷启动耗时长)，甚至长达数分钟，在这种场景下，从按需启动应用容器到完成应用启动开始响应请求，过程耗时很长，没办法满足应用响应时间的性能要求，如何减少从唤醒触发到启动结束的这个时间间隔，成为一个难题；

[0013] 本专利申请提出的容器内存快照技术，实现的serverless应用快速唤醒的方法和系统，该方法是指一种基于用户空间进程快照的技术(CRIU)，通过对容器进行内存级快照，极大加快应用从“休眠”(0个实例)到“唤醒”(1个以上实例)的速度，使得资源在应用空闲时得到最大化的释放，并且能让应用具备快速唤醒的响应能力；

[0014] CRIU是Checkpoint/Restore In Userspace技术的缩写，是Linux中开源通用的一种进程快照技术，它分为快照保存和快照恢复两个阶段，CRIU的独特之处在于它在用户空间而不是内核中实现；

[0015] 快照保存：利用CRIU，可以冻结正在运行的应用程序，并将其作为文件的集合检查点到硬盘中；

[0016] 快照恢复：可以从冻结点开始，使用快照文件来还原和运行该应用程序；

[0017] 无服务场景中会有容器管理引擎负责容器的启动、停止以及快照创建和快照恢复等，容器管理引擎内置了容器快照管理能力，能够通过CRIU技术，实现容器的快照创建和快照恢复，达到从容器快照恢复容器时，其文件系统和内存数据，进程数据都是跟快照创建时一模一样的效果。

发明内容

[0018] 本发明要解决的技术问题是在serverless的架构下 “缩容为0，再进行按需唤醒”的环节中如何减少从唤醒触发到启动结束的这个时间间隔，本发明提供一种实现容器休眠与唤醒的系统，能够使得“缩容为0，再进行按需唤醒”更加适用于各种不同的应用类型，使得不再为应用冷启动时间长而苦恼， 用以解决现有技术导致的缺陷。

[0019] 本发明还提供一种实现容器休眠的方法；

[0020] 本发明还提供一种实现容器唤醒的方法；

[0021] 为解决上述技术问题本发明提供以下的技术方案：

[0022] 第一方面，一种实现容器休眠与唤醒的系统，其中，包含容器引擎、网关模块、快照管理模块、休眠管理模块以及一个或多个应用，所述应用由一个或多个容器组成；

[0023] 所述网关模块用于接收流量请求并根据所述流量请求生成访问请求数据并对所述应用进行流量的接入、转发以及生成唤醒请求并将所述唤醒请求与所述访问请求数据传输至所述休眠管理模块；

[0024] 所述休眠管理模块用于与所述网关模块创建连接获取所述访问请求数据，并判断所述应用是否需要进入休眠状态并生成休眠指令，还用于传输所述休眠指令至所述快照管理模块，还用于接收所述唤醒请求并依据所述唤醒请求生成传输至所述快照管理模块的恢

复指令，还用于与所述网关模块创建连接同步所述应用的所述休眠指令与所述恢复指令；

[0025] 所述快照管理模块用于接收、管理快照，还用于维护所述快照与所述容器的对应关系，还用于接收所述休眠指令并根据所述休眠指令对所述容器进行休眠，还用于接收所述恢复指令对所述容器进行恢复，所述快照管理模块还用于触发所述容器引擎工作；

[0026] 所述容器引擎通过CRIU对所述容器进行快照的创建、恢复，并将所述快照传输至所述快照管理模块，一般来说所述容器引擎有多种，但是这些所述容器引擎有同样的一套标准接口，所述快照管理模块调用所述容器引擎的标准接口进行快照的创建、恢复等。

[0027] 上述的一种实现容器休眠与唤醒的系统，其中，所述网关模块内置有流量识别模块；

[0028] 所述流量识别模块用于识别所述流量请求确定目标应用，所述网关模块将所述流量转发至所述目标应用。

[0029] 上述的一种实现容器休眠与唤醒的系统，其中，所述休眠管理模块内置有过期时间管理模块；

[0030] 所述过期时间管理模块用于记录所述应用在设定的时间内接收到的请求，并生成所述休眠指令。

[0031] 上述的一种实现容器休眠与唤醒的系统，其中，所述快照管理模块内置有关系表创建模块；

[0032] 所述关系表创建模块用于维护所述快照与所述容器的对应关系并生成关系表；

[0033] 所述快照管理模块接收到所述休眠指令时还用于查询所述容器所在的主机，并调用所述容器所在主机的所述容器引擎将所述容器利用CRIU保存成快照。

[0034] 上述的一种实现容器休眠与唤醒的系统，其中，所述休眠管理模块内置有快照恢复模块与过期时间重置模块；

[0035] 所述快照恢复模块用于接收所述恢复指令时对相应的所述快照中恢复所述容器；

[0036] 所述过期时间重置模块用于所述容器恢复后对该容器进行过期时间的重置；

[0037] 所述休眠管理模块还用于所述容器的状态传输至所述网关模块，所述网关模块将接收到的上述请求转发至唤醒后的所述容器中。

[0038] 上述的一种实现容器休眠与唤醒的系统，其中，所述快照恢复模块根据所述容器的编号或名称确定所述快照后调用所述容器引擎的API进行所述快照的恢复。

[0039] 第二方面，一种实现容器休眠的方法，其中，包含以下步骤：

[0040] 步骤A1：流量进入网关模块后网关模块根据流量请求的信息识别出流量转发的目标应用，网关模块将请求转发至目标应用的容器中；

[0041] 步骤A2：休眠管理模块根据从网关模块获取的访问请求数据与休眠策略结合判断是否有应用需要休眠，并生成传输至快照管理模块的休眠指令，休眠管理模块与网关模块连接进行休眠指令的同步，网关模块相当于外部访问的入口，访问请求数据来自于系统外的客户端(比如手机浏览器、PC端等所有的外部访问)；

[0042] 步骤A3：快照管理模块接收休眠指令并依据该休眠指令对相应的容器进行休眠，同时快照管理模块从容器管理平台查询该容器所在的主机并调用该主机的容器引擎将该容器利用CRIU技术保存成快照，容器管理平台为Serverless 系统所运行容器的管理平台，就是附图中的“serverless 框架”；

[0043] 步骤A4:快照管理模块维护快照与容器之间对应关系的同时生成关系表,关系表的内容主要是指快照和应用容器的对应关系,比如 A 应用需要休眠,那么创建快照A1,然后A应用需要恢复的时候,需要找到 A1 进行还原,而一个系统或者无服务的管理平台中,会有很多应用,这时候就需要维护应用容器与快照之间的映射关系,并根据这个关系去还原正确的快照;对于无服务框架来说,扩容缩容都是针对应用的,应用流量大那么就需要创建更多的容器,当本技术方案采用1个容器实例的时候还需要缩容,那么需要给这个应用创建一个快照,这个快照是提供应用和容器实例对应关系,应用和容器之间的关系也是需要系统维护的,但是针对本技术方案,只需要确定容器和快照的关系即可;

[0044] 在整个系统平台的角度,还会有很多其他的数据表,比如网关模块需要维护应用的访问路径和访问地址的关系表。

[0045] 上述的一种实现容器休眠的方法,其中,步骤2中判断是否有应用需要休眠的方法为:

[0046] 获取应用在设定范围内的任何请求并统计个数,当请求个数为0时,则生成休眠指令。

[0047] 网关模块是平台上所有应用的流量入口,流量进入网关模块后,网关模块会根据流量请求的信息,比如请求的URL目标地址,识别出流量转发的目标应用,当目标应用存在运行的容器实例时,网关模块会把请求正常转发到对应的实例上;休眠管理模块从网关模块获得应用的访问请求数据,并根据一定的休眠策略判断是否有应用需要被休眠:假设休眠策略是基于请求活跃度的,当一个应用在设定的时间范围内没有任何请求,则进行休眠,请注意,本技术方案不限于上述休眠策略;休眠管理模块维护一个应用访问的过期时间表,当应用被访问的时候,该应用的过期时间会被重置;当休眠管理模块发现应用过期时,休眠管理模块会触发快照管理模块休眠该应用,同时,休眠管理模块会通知网关模块,对应应用进入休眠状态;快照管理模块会维护一张应用容器实例和快照的关系表,当接收到某个应用的休眠指令时,快照管理模块会从容器管理平台查询该应用容器所在的主机,并调用应用容器所在主机的容器引擎将应用容器实例利用CRIU技术保存成快照。

[0048] 第三方面,一种实现容器唤醒的方法,其中,包含以下步骤:

[0049] 步骤B1:当网关模块接收到休眠容器的请求时,网关模块保持该请求并生成发送至休眠管理模块唤醒请求;

[0050] 步骤B2:休眠管理模块接收到唤醒请求后生成传输至快照管理模块的恢复请求;

[0051] 步骤B3:快照管理模块依据恢复请求对相应的容器进行恢复,并重置容器的过期时间;

[0052] 步骤B4:快照管理模块根据容器的编号或名称确认快照ID并调用容器引擎的API恢复快照;

[0053] 步骤B5:容器从快照恢复完毕后,休眠管理模块与网关模块连接同步容器的状态,网关模块将请求转发给容器。

[0054] 当网关模块接收到休眠应用对应的请求时,网关模块会保持该请求,并给休眠管理模块发出唤醒应用的请求;休眠管理模块触发快照管理模块从应用快照中恢复应用,并在快照恢复成功后将该应用的过期时间重置;快照管理模块会根据应用编号或者名称确认快照 ID,调用容器引擎 API 恢复快照;当应用从快照恢复完毕后,休眠管理模块向网关模

块标示应用状态，之后网关会将请求转发给应用实例。

[0055] 第四方面，一种系统处理装置，其中，包含至少一个处理器与所述至少一个处理器耦合的存储器；

[0056] 所述存储器存储有可执行指令，其中，所述可执行指令在被所述至少一个处理器执行时使得实现上述方法中任一项所述的方法。

[0057] 第五方面，一种计算机可读存储介质，其特征在于，所述计算机可读存储介质上存储有计算机程序，所述计算机程序被处理器执行时实现上述中任一项所述的方法的步骤。

[0058] 依据上述本发明一种实现容器休眠与唤醒的系统及休眠与唤醒方法的技术方案具有以下技术效果：

[0059] 在Serverless场景下，通过进程快照技术，极大减少了某些应用“唤醒”的时间，从之前几分钟减少到1-2秒以内，解决了某些应用因为“冷启动”时间长，而无法实现serverless场景的“无请求即无需占有资源，有请求按需唤醒”的要求，该方法还可以延伸到其他场景，比如在应用有突发流量的情况下，同样需要容器快速拉起快速响应的需求(从N到N+1)，同样可以利用该方法和系统来满足这个需求。

附图说明

[0060] 图1为本发明一种实现容器休眠与唤醒的系统的结构示意图；

[0061] 图2为本发明一种实现容器休眠的方法的流程图；

[0062] 图3为本发明一种实现容器唤醒的方法的流程图；

[0063] 图4为serverless框架的第一种实例状态结构示意图；

[0064] 图5为serverless框架的第二种实例状态结构示意图；

[0065] 图6为serverless框架的第三种实例状态结构示意图；

[0066] 图7为serverless框架的第四种实例状态结构示意图；

[0067] 图8为serverless框架中基本的唤醒过程结构示意图。

具体实施方式

[0068] 为了使发明实现的技术手段、创造特征、达成目的和功效易于明白了解，下结合具体图示，对本发明实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例是本发明的一部分实施例，而不是全部的实施例。

[0069] 基于本发明中的实施例，本领域普通技术人员在没有做出创造性劳动的前提下所获得的所有其他实施例，都属于本发明保护的范围。

[0070] 须知，本说明书所附图式所绘示的结构、比例、大小等，均仅用以配合说明书所揭示的内容，以供熟悉此技术的人士了解与阅读，并非用以限定本发明可实施的限定条件，故不具技术上的实质意义，任何结构的修饰、比例关系的改变或大小的调整，在不影响本发明所能产生的功效及所能达成的目的下，均应仍落在本发明所揭示的技术内容得能涵盖的范围内。

[0071] 同时，本说明书中所引用的如“上”、“下”、“左”、“右”、“中间”及“一”等的用语，亦仅为便于叙述的明了，而非用以限定本发明可实施的范围，其相对关系的改变或调整，在无实质变更技术内容下，当亦视为本发明可实施的范畴。

[0072] 本发明的一较佳实施例是提供一种实现容器休眠与唤醒的系统及休眠与唤醒方法，目的是在Serverless场景下，通过进程快照技术，极大减少了某些应用“唤醒”的时间，从之前几分钟减少到1-2秒以内，解决了某些应用因为“冷启动”时间长，而无法实现serverless场景的“无请求即无需占有资源，有请求按需唤醒”的要求，该方法还可以延伸到其他场景，比如在应用有突发流量的情况下，同样需要容器快速拉起快速响应的需求(从N到N+1)，同样可以利用该方法和系统来满足这个需求。

[0073] 如图1所示，第一方面，一种实现容器休眠与唤醒的系统，其中，包含容器引擎、网关模块、快照管理模块、休眠管理模块以及一个或多个应用，应用由一个或多个容器组成；

[0074] 网关模块用于接收流量请求并根据流量请求生成访问请求数据并对应用进行流量的接入、转发以及生成唤醒请求并将唤醒请求与访问请求数据传输至休眠管理模块；

[0075] 休眠管理模块用于与网关模块创建连接获取访问请求数据，并判断应用是否需要进入休眠状态并生成休眠指令，还用于传输休眠指令至快照管理模块，还用于接收唤醒请求并依据唤醒请求生成传输至快照管理模块的恢复指令，还用于与网关模块创建连接同步应用的休眠指令与恢复指令；

[0076] 快照管理模块用于接收、管理快照，还用于维护快照与容器的对应关系，还用于接收休眠指令并根据休眠指令对容器进行休眠，还用于接收恢复指令对容器进行恢复，快照管理模块还用于触发容器引擎工作；

[0077] 容器引擎通过CRIU对容器进行快照的创建、恢复，并将快照传输至快照管理模块，一般来说容器引擎有多种，但是这些容器引擎有同样的一套标准接口，快照管理模块调用容器引擎的标准接口进行快照的创建、恢复等。

[0078] 其中，网关模块内置有流量识别模块；

[0079] 流量识别模块用于识别流量请求确定目标应用，网关模块将流量转发至目标应用。

[0080] 其中，休眠管理模块内置有过期时间管理模块；

[0081] 过期时间管理模块用于记录应用在设定的时间内接收到的请求，并生成休眠指令。

[0082] 其中，快照管理模块内置有关系表创建模块；

[0083] 关系表创建模块用于维护快照与容器的对应关系并生成关系表；

[0084] 快照管理模块接收到休眠指令时还用于查询容器所在的主机，并调用容器所在主机的容器引擎将容器利用CRIU保存成快照。

[0085] 其中，休眠管理模块内置有快照恢复模块与过期时间重置模块；

[0086] 快照恢复模块用于接收恢复指令时对相应的快照中恢复容器；

[0087] 过期时间重置模块用于容器恢复后对该容器进行过期时间的重置；

[0088] 休眠管理模块还用于容器的状态传输至网关模块，网关模块将接收到的上述请求转发至唤醒后的容器中。

[0089] 其中，快照恢复模块根据容器的编号或名称确定快照后调用容器引擎的API进行快照的恢复。

[0090] 如图2所示，第二方面，一种实现容器休眠的方法，其中，包含以下步骤：

[0091] 步骤A1：流量进入网关模块后网关模块根据流量请求的信息识别出流量转发的目

标应用,网关模块将请求转发至目标应用的容器中;

[0092] 步骤A2:休眠管理模块根据从网关模块获取的访问请求数据与休眠策略结合判断是否有应用需要休眠,并生成传输至快照管理模块的休眠指令,休眠管理模块与网关模块连接进行休眠指令的同步,网关模块相当于外部访问的入口,访问请求数据来自于系统外的客户端(比如手机浏览器、PC端等所有的外部访问),比如API网关,API网关是一个服务器,是系统的唯一入口,API网关封装了系统内部架构,为每个客户端提供一个定制的API,它可能还具有其它职责,如身份验证、监控、负载均衡、缓存、请求分片与管理、静态响应处理;API网关方式的核心要点是,所有的客户端和消费端都通过统一的网关接入微服务,在网关层处理所有的非业务功能,通常,网关也是提供REST/HTTP的访问API,服务端通过API-GW注册和管理服务;

[0093] 步骤A3:快照管理模块接收休眠指令并依据该休眠指令对相应的容器进行休眠,同时快照管理模块从容器管理平台查询该容器所在的主机并调用该主机的容器引擎将该容器利用CRIU技术保存成快照,容器管理平台为Serverless 系统所运行容器的管理平台,就是附图中的“serverless 框架”;

[0094] 步骤A4:快照管理模块维护快照与容器之间对应关系的同时生成关系表,关系表的内容主要是指快照和应用容器的对应关系,比如 A 应用需要休眠,那么创建快照A1,然后A应用需要恢复的时候,需要找到 A1 进行还原,而一个系统或者无服务的管理平台中,会有很多应用,这时候就需要维护应用容器与快照之间的映射关系,并根据这个关系去还原正确的快照;对于无服务框架来说,扩容缩容都是针对应用的,应用流量大那么就需要创建更多的容器,当本技术方案采用1个容器实例的时候还需要缩容,那么需要给这个应用创建一个快照,这个快照是提供应用和容器实例对应关系,应用和容器之间的关系也是需要系统维护的,但是针对本技术方案,只需要确定容器和快照的关系即可;

[0095] 在整个系统平台的角度,还会有很多其他的数据表,比如网关模块需要维护应用的访问路径和访问地址的关系表。

[0096] 其中,步骤2中判断是否有应用需要休眠的方法为:

[0097] 获取应用在设定范围内的任何请求并统计个数,当请求个数为0时,则生成休眠指令。

[0098] 网关模块是平台上所有应用的流量入口,流量进入网关模块后,网关模块会根据流量请求的信息,比如请求的URL目标地址,识别出流量转发的目标应用,当目标应用存在运行的容器实例时,网关模块会把请求正常转发到对应的实例上;休眠管理模块从网关模块获得应用的访问请求数据,并根据一定的休眠策略判断是否有应用需要被休眠:假设休眠策略是基于请求活跃度的,当一个应用在设定的时间范围内没有任何请求,则进行休眠,请注意,本技术方案不限于上述休眠策略;休眠管理模块维护一个应用访问的过期时间表,当应用被访问的时候,该应用的过期时间会被重置;当休眠管理模块发现应用过期时,休眠管理模块会触发快照管理模块休眠该应用,同时,休眠管理模块会通知网关模块,对应应用进入休眠状态;快照管理模块会维护一张应用容器实例和快照的关系表,当接收到某个应用的休眠指令时,快照管理模块会从容器管理平台查询该应用容器所在的主机,并调用应用容器所在主机的容器引擎将应用容器实例利用CRIU技术保存成快照。

[0099] 如图3所示,第三方面,一种实现容器唤醒的方法,其中,包含以下步骤:

[0100] 步骤B1:当网关模块接收到休眠容器的请求时,网关模块保持该请求并生成发送至休眠管理模块唤醒请求;

[0101] 步骤B2:休眠管理模块接收到唤醒请求后生成传输至快照管理模块的恢复请求;

[0102] 步骤B3:快照管理模块依据恢复请求对相应的容器进行恢复,并重置容器的过期时间;

[0103] 步骤B4:快照管理模块根据容器的编号或名称确认快照ID并调用容器引擎的API恢复快照;

[0104] 步骤B5:容器从快照恢复完毕后,休眠管理模块与网关模块连接同步容器的状态,网关模块将请求转发给容器。

[0105] 当网关模块接收到休眠应用对应的请求时,网关模块会保持该请求,并给休眠管理模块发出唤醒应用的请求;休眠管理模块触发快照管理模块从应用快照中恢复应用,并在快照恢复成功后将该应用的过期时间重置;快照管理模块会根据应用编号或者名称确认快照 ID,调用容器引擎 API 恢复快照;当应用从快照恢复完毕后,休眠管理模块向网关模块标示应用状态,之后网关会将请求转发给应用实例。

[0106] 第四方面,一种系统处理装置,其中,包含至少一个处理器与至少一个处理器耦合的存储器;

[0107] 存储器存储有可执行指令,其中,可执行指令在被至少一个处理器执行时使得实现上述方法中任一项的方法。

[0108] 第五方面,一种计算机可读存储介质,其特征在于,计算机可读存储介质上存储有计算机程序,计算机程序被处理器执行时实现上述中任一项的方法的步骤。

[0109] 例如,存储器可以包括随机存储器、闪存、只读存储器、可编程只读存储器、非易失性存储器或寄存器等;

[0110] 处理器可以是中央处理器(Central Processing Unit,CPU)等,或者是图像处理器 (Graphic Processing Unit,GPU)存储器可以存储可执行指令;

[0111] 处理器可以执行在存储器中存储的执行指令,从而实现本文描述的各个过程。

[0112] 可以理解,本实施例中的存储器可以是易失性存储器或非易失性存储器,或可包括易失性和非易失性存储器两者;

[0113] 其中,非易失性存储器可以是ROM (Read-OnlyMemory,只读存储器)、PROM (ProgrammableROM,可编程只读存储器)、EPROM (ErasablePROM,可擦除可编程只读存储器)、EEPROM (ElectricallyEPROM,电可擦除可编程只读存储器)或闪存。

[0114] 易失性存储器可以是RAM (RandomAccessMemory,随机存取存储器),其用作外部高速缓存;

[0115] 通过示例性但不是限制性说明,许多形式的RAM可用,例如SRAM (StaticRAM,静态随机存取存储器)、DRAM (DynamicRAM,动态随机存取存储器)、SDRAM (SynchronousDRAM,同步动态随机存取存储器)、DDRSDRAM (DoubleDataRate SDRAM,双倍数据速率同步动态随机存取存储器)、ESDRAM (Enhanced SDRAM,增强型同步动态随机存取存储器)、SLDRAM (SynchlinkDRAM,同步连接动态随机存取存储器)和DRRAM (DirectRambusRAM,直接内存总线随机存取存储器)。本文描述的存储器205旨在包括但不限于这些和任意其它适合类型的存储器205。

[0116] 在一些实施方式中，存储器存储了如下的元素，升级包、可执行单元或者数据结构，或者他们的子集，或者他们的扩展集：操作系统和应用程序；

[0117] 其中，操作系统，包含各种系统程序，例如框架层、核心库层、驱动层等，用于实现各种基础业务以及处理基于硬件的任务；

[0118] 应用程序，包含各种应用程序，用于实现各种应用业务。实现本发明实施例方法的程序可以包含在应用程序中。

[0119] 本领域技术人员可以明白的是，结合本文中所公开的实施例描述的各示例的单元及算法步骤能够以电子硬件、或者软件和电子硬件的结合来实现；

[0120] 这些功能是以硬件还是软件方式来实现，取决于技术方案的特定应用和设计约束条件；

[0121] 本领域技术人员可以针对每个特定的应用，使用不同的方式来实现所描述的功能，但是这种实现并不应认为超出本申请的范围。

[0122] 在本申请实施例中，所公开的系统、装置和方法可以通过其它方式来实现；

[0123] 例如，单元或模块的划分仅仅为一种逻辑功能划分，在实际实现时还可以有另外的划分方式；

[0124] 例如，多个单元或模块或组件可以进行组合或者可以集成到另一个系统中；

[0125] 另外，在本申请实施例中的各功能单元或模块可以集成在一个处理单元或模块中，也可以是单独的物理存在等等。

[0126] 应理解，在本申请的各种实施例中，各过程的序号的大小并不意味着执行顺序的先后，各过程的执行顺序应以其功能和内在逻辑确定，而不应对本申请的实施例的实施过程构成任何限定。

[0127] 所述功能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，可以存储在机器可读存储介质中；

[0128] 因此，本申请的技术方案可以以软件产品的形式来体现，该软件产品可以存储在机器可读存储介质中，其可以包括若干指令用以使得电子设备执行本申请实施例所描述的技术方案的全部或部分过程；

[0129] 上述存储介质可以包括ROM、RAM、可移动盘、硬盘、磁盘或者光盘等各种可以存储程序代码的介质。

[0130] 综上，本发明的一种实现容器休眠与唤醒的系统及休眠与唤醒方法，能够在Serverless场景下，通过进程快照技术，极大减少了某些应用“唤醒”的时间，从之前几分钟减少到1-2秒以内，解决了某些应用因为“冷启动”时间长，而无法实现serverless场景的“无请求即无需占有资源，有请求按需唤醒”的要求，该方法还可以延伸到其他场景，比如在应用有突发流量的情况下，同样需要容器快速拉起快速响应的需求(从N到N+1)，同样可以利用该方法和系统来满足这个需求。

[0131] 以上对发明的具体实施例进行了描述。需要理解的是，发明并不局限于上述特定实施方式，其中未尽详细描述的设备和结构应该理解为用本领域中的普通方式予以实施；本领域技术人员可以在权利要求的范围内做出各种变形或修改做出若干简单推演、变形或替换，这并不影响发明的实质内容。

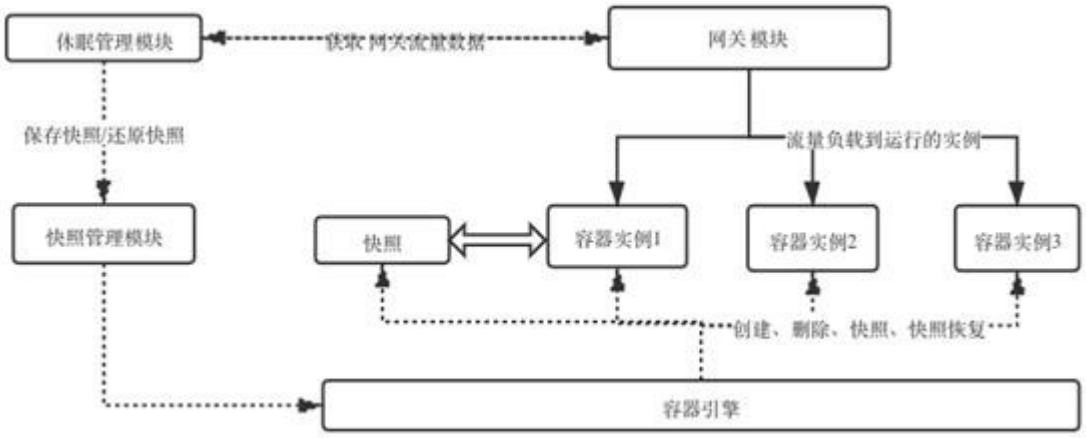

休眠管理模块
获取网关流量数据
网关模块
保存快照/还原快照
流量负载到运行的实例
快照管理模块
快照
容器实例1
容器实例2
容器实例3
创建、删除、快照、快照恢复
容器引擎

图1

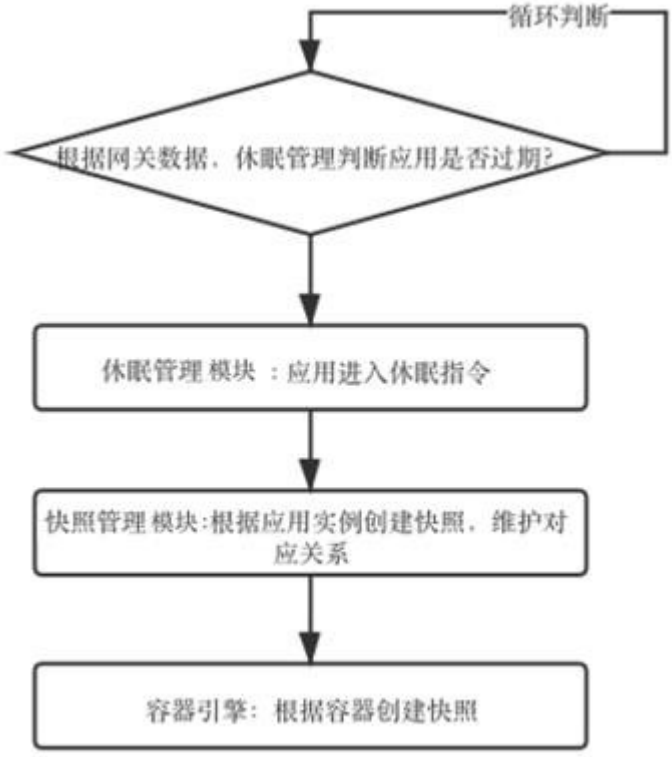

循环判断
根据网关数据，休眠管理判断应用是否过期?
休眠管理模块 ：应用进入休眠指令
快照管理模块:根据应用实例创建快照，维护对应关系
容器引擎：根据容器创建快照

图2

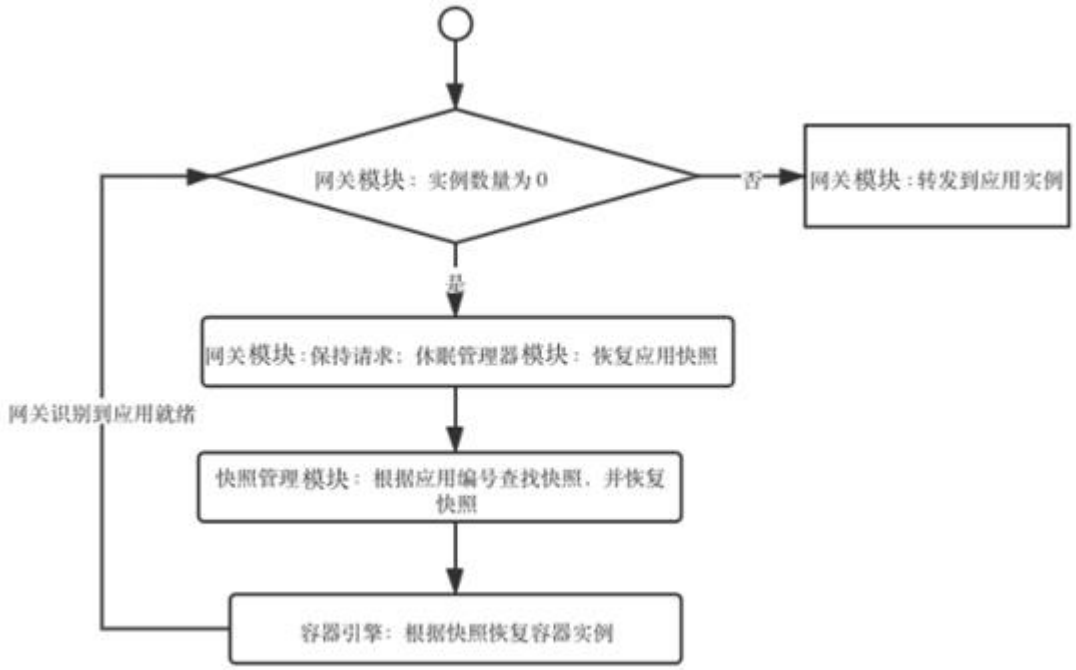

网关模块：实例数量为0
否
网关模块：转发到应用实例
是
网关模块：保持请求；休眠管理器模块：恢复应用快照
网关识别到应用就绪
快照管理模块：根据应用编号查找快照，并恢复快照
容器引擎：根据快照恢复容器实例

图3

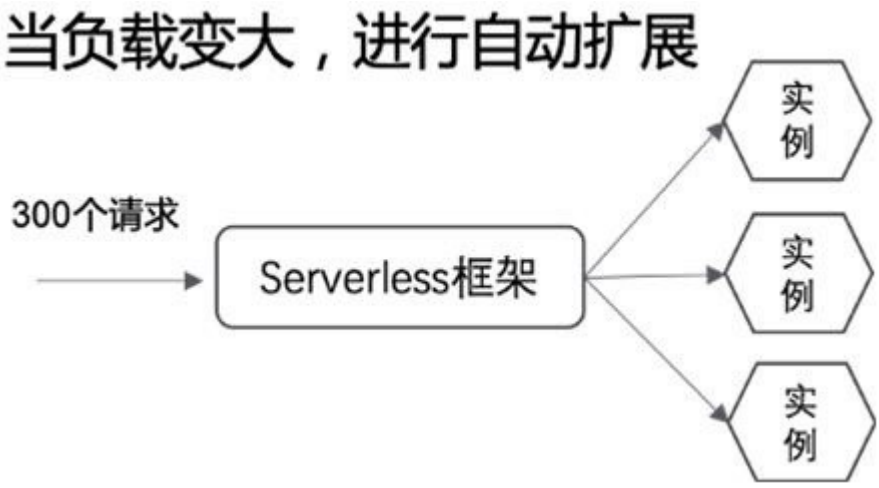

当负载变大，进行自动扩展
300个请求
Serverless框架
实例
实例
实例

图4


100个请求
Serverless框架
实例

图5


0个请求
Serverless框架

图6

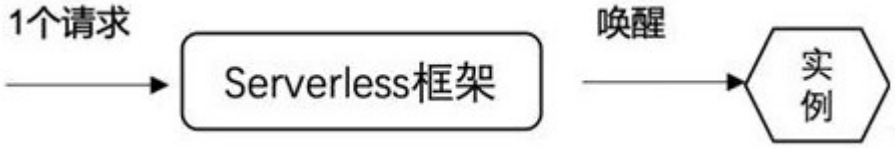

1个请求
Serverless框架
唤醒
实例

图7

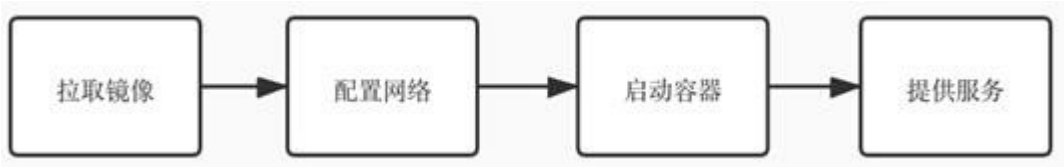

拉取镜像
配置网络
启动容器
提供服务

图8