基于容器的应用管理方法和装置

本申请要求 2020 年 10 月 30 日递交的申请号为 202011194163.X、发明名称为“基于容器的应用管理方法和装置”的中国专利申请的优先权，其全部内容通过引用结合在本申请中。

技术领域

本公开涉及一种无服务器计算领域，特别涉及基于容器的应用管理方案。

背景技术

在云计算场景中，无服务器计算（Serverless）是一种模式，使应用开发人员能专注于他们的核心产品，而无需关心应用程序在本地或云端的服务器运行状态。

在无服务器（Serverless）计算领域的自动扩缩容场景中，应用程序的弹性时间，即应用程序实现弹性伸缩所需要的时间，是一个非常重要的指标。

特别是在云计算场景中，需要根据应用程序的实时流量对应用实例进行弹性伸缩。如果一个应用的弹性时间过长，将导致调度平台无法灵活快速地对实例进行扩容和缩容。或者，扩容和缩容的流量监测需要经过很长的时间才能保证缩容对服务没有影响，从而导致计算资源被浪费。

在扩容时，需要实现实例的快速启动。目前部分系统或者环境下为了实现实例的快速启动，主要有下面两种方案。

一种方案是借助应用架构，使用分叉（Fork）或者克隆（Clone）技术来创建新的进程，从而加速应用冷启动。以安卓（Android）移动操作系统为例，其系统中应用启动普遍使用的受精卵（Zygote）框架，其它运行实例的创建均是通过从受精卵（Zygote）主进程分叉（Fork）而来。尽可能在分叉（Fork）之前加载应用所依赖的相同服务，从而达到应用快速启动。

基于 Zygote 模型的应用架构的扩容方案非常依赖这种应用架构。对于存量应用，需要对应用架构进行大幅调整。因此，这种方案不适用于存量应用。

而且，通常而言，应用冷启动的耗时较长，也无法满足快速扩容的要求。

另一种方案是利用检查点/恢复（Checkpoint/Restore）快照技术。应用启动之后制作成快照。应用要扩容的时候，利用已经制作好的快照，来加速应用的启动过程。

快照方案通常又有两种。

一种是快照复制方案，针对一个应用只制作一个快照。利用该快照来创建所有的应用实例。

使用这种快照复制方式，如果制作快照的时候，应用中存在一些持久化状态信息，例如 UUID 信息等，那么所生成的多个实例就会在正确性和安全性存在问题。

另一种是一对一快照方式，一个应用实例对应于一个快照，一个快照用于恢复一个应用实例。

这种快照方式需要提前启动应用实例，然后针对已启动的应用实例制作快照予以存储。需要恢复该应用实例的时候再恢复对应的快照。因此，这种方案的快照管理和存储的成本较高。

而且，快照制作过程中可能会和具体的机器特征绑定，导致后续的快照恢复仍然需要在具有所绑定的机器特征的机器上进行，不能在其它环境进行。因此，快照不通用。

另外，一对一快照方式还存在时间状态问题。例如，对于一些依赖于实际时间的应用，制作快照时会依赖于当时的时间；恢复快照时也会同时恢复制作快照时的时间。这样，应用逻辑可能会受时间影响而导致发生执行逻辑错误。

换言之，在现有技术中，在无服务器计算领域，应用的弹性扩容和缩容能力尚不能够满足日益增长的应用弹性需求。

因此，仍然需要一种能够例如随着应用程序的流量变化而对应用实例进行快速弹性伸缩的方案。

发明内容

本公开要解决的一个技术问题是提供一种基于容器的应用管理方法和装置，其能够在降低应用实例成本的同时，对应用实例进行快速弹性伸缩。

根据本公开的第一个方面，提供了一种基于容器的应用管理方法，包括：配置基于容器实现的无服务器计算系统，以使得允许应用实例在运行时处于在线状态和低功耗状态之一，其中，应用实例在低功耗状态下的功耗和/或资源消耗低于在线状态下的功耗和/或资源消耗；响应于对应用执行缩容处理，使应用的至少一个在线状态的第一应用实例进入低功耗状态；以及响应于对应用执行扩容处理，使应用的至少一个低功耗状态的第二应用实例进入在线状态。

可选地，使应用的至少一个在线状态的第一应用实例进入低功耗状态的步骤包括：限制至少一个第一应用实例的资源使用额度；以及降低至少一个第一应用实例各自所在容器的资源配置，并且/或者，使应用的至少一个低功耗状态的第二应用实例进入在线状态的步骤包括：提升至少一个第二应用实例各自所在容器的资源配置；以及解除至少一个第二应用实例的资源使用额度限制。

可选地，使应用的至少一个低功耗状态的第二应用实例进入在线状态的步骤包括：优先选择使容器所在节点中资源相对空闲的机器上的容器中的第二应用实例进入在线状态。

可选地，该方法还可以包括：在应用实例将要进入在线状态而其所在节点上资源不足的情况下，基于用户态的检查点恢复（CRIU）技术的实时迁移功能，将应用实例迁移到相对空闲的节点上；以及/或者在同一个节点上具有多个低功耗状态的应用实例的情况下，基于用户态的检查点恢复（CRIU）技术，将一个或多个应用实例做成检查点快照，然后基于检查点快照在相对空闲的节点上恢复一个或多个应用实例。

可选地，基于控制组机制来限制应用实例的资源使用额度或解除限制；并且/或者在降低容器的资源配置时，基于容器荚资源原地升级机制，将应用实例所在容器所释放的资源归还给调度系统；以及/或者在提升容器的资源配置时，基于容器荚资源原地升级机制，从调度系统为容器申请资源。

可选地，使至少一个第一应用实例进入低功耗状态的步骤包括：基于 CPU 共享功能，将多个第一应用实例加入低功耗规格的 CPU 组中运行，低功耗规格的 CPU 组中的应用实例共享 CPU；并且/或者使至少一个第二应用实例进入在线状态的步骤包括：使第二应用实例退出低功耗规格的 CPU 组。

可选地，使至少一个第一应用实例进入低功耗状态的步骤包括：使至少一个第一应用实例所占用的内存空间处于内存中的低功耗实例内存范围中；并且/或者使至少一个第二应用实例进入在线状态的步骤包括：使至少一个第二应用实例所占用的内存空间处于内存中的低功耗实例内存范围之外。

可选地，低功耗实例内存范围是内存中周期性内存管理操作在内存中的执行范围之外的范围。

可选地，周期性内存管理操作包括内存垃圾回收操作和/或释放预定期限内未曾使用过的内存的内存释放操作。

可选地，该方法还可以包括：响应于范围调整指令，调整执行范围和/或执行范围的大小；以及/或者在占用内存空间的第一应用实例进入低功耗状态的情况下，设置执行范围不包括第一应用实例所占用的内存空间；以及/或者在占用内存空间的第一应用实例进入在线状态的情况下，设置执行范围包括第一应用实例所占用的内存空间。

可选地，使至少一个第一应用实例进入低功耗状态的步骤可以包括：关闭第一应用实例所使用的部分资源或降低部分资源的使用量，仅保留第一应用实例所使用的部分系统资源。

可选地，该方法还可以包括：使一个或多个低功耗状态的应用实例的内存数据在内存和存储设备之间转移。

可选地，使一个或多个低功耗应用实例的内存数据在内存和存储设备之间转移的步骤包括：使用系统内核态的内存交换功能，使一个或多个低功耗应用实例的内存数据在内存和存储设备之间转移；以及/或者使用用户态的内存交换功能，使一个或多个低功耗应用实例的内存数据在内存和存储设备之间转移，并且将来自不同容器的内存数据转移到不同的存储设备或存储设备的不同页面。

可选地，使用用户态的内存交换功能，同时将多个低功耗应用实例的内存数据从内存分别转移到不同的存储设备或存储设备的不同页面；并且/或者，同时将多个低功耗应用实例的内存数据从不同的存储设备或存储设备的不同页面转移到内存中。

可选地，该方法还可以包括：接收针对应用或应用实例的内存交换设置指令，内存

交换设置指令用于指示设置针对该应用的应用实例或该应用实例，使用系统内核态的内存交换功能还是用户态的内存交换功能来执行内存数据转移；响应于内存交换设置指令，针对该应用的应用实例或该应用实例，设置用于执行内存数据转移的内存交换功能。

可选地，在使用用户态的内存交换功能的情况下，使用多个超性能云盘设备构造具有相同优先级的内存交换用存储设备。

可选地，一个或多个低功耗状态的应用实例的内存数据在内存和存储设备之间转移的步骤包括：基于最近最少使用算法，选择一个或多个低功耗状态的应用实例的内存数据，并将其从内存转移并持久化到存储设备；以及/或者响应于换回指令或流量请求或实例部署策略变化，将位于存储设备上的一个或多个低功耗状态的应用实例的内存数据转移回到内存中。

可选地，在第二应用实例的内存数据已被转移到存储设备上的情况下，使应用的至少一个低功耗状态的第二应用实例进入在线状态的步骤还包括：将第二应用实例的内存数据从存储设备转移到内存中周期性内存管理操作的执行范围之中。

可选地，响应于针对应用的缩容指令，或响应于应用的流量降低到低于第一预定阈值而成为低流量应用，对应用执行缩容处理；并且/或者响应于针对应用的扩容指令，或响应于应用的流量增加到高于第二预定阈值，对应用执行扩容处理。

根据本公开的第二个方面，提供了一种基于容器的应用管理装置，部署于基于容器实现的无服务器计算系统中，无服务器计算系统和被配置为允许应用实例在运行时处于在线状态和低功耗状态之一，其中，应用实例在低功耗状态下的功耗和/或资源消耗低于在线状态下的功耗和/或资源消耗，该装置包括：缩容装置，响应于对应用执行缩容处理，使应用的至少一个在线状态的第一应用实例进入低功耗状态；以及扩容装置，响应于对应用执行扩容处理，使应用的至少一个低功耗状态的第二应用实例进入在线状态。

可选地，缩容装置可以是包括：额度限制装置，用于限制低流量应用的至少一个第一应用实例的资源使用额度；以及配置降低装置，用于降低至少一个第一应用实例各自所在容器的资源配置。

可选地，扩容装置可以包括：配置提升装置，用于提升至少一个第一应用实例中至少一个第二应用实例各自所在容器的资源配置；以及额度恢复装置，用于解除至少一个第二应用实例的资源使用额度限制。

可选地，该装置还可以包括：内存交换存储装置，使至少一个第一应用实例中的一个或多个第一应用实例的内存数据在内存和存储设备之间转移。

根据本公开的第三个方面，提供了一种计算设备，包括：处理器；以及存储器，其上存储有可执行代码，当可执行代码被处理器执行时，使处理器执行如上述第一方面所述的方法。

根据本公开的第四个方面，提供了一种非暂时性机器可读存储介质，其上存储有可

执行代码，当可执行代码被电子设备的处理器执行时，使处理器执行如上述第一方面所述的方法。

由此，实现了一种能够对应用实例进行快速弹性伸缩的基于容器的应用管理方法和装置。

附图说明

通过结合附图对本公开示例性实施方式进行更详细的描述，本公开的上述以及其它目的、特征和优势将变得更加明显，其中，在本公开示例性实施方式中，相同的参考标号通常代表相同部件。

图 1 是根据本公开实施例的应用弹性缩容的示意图；

图 2 是根据本公开实施例的应用弹性扩容的示意图；

图 3 是根据本公开实施例的应用管理装置的示意性框图；

图 4 是根据本公开实施例的应用管理方法的示意性流程图；

图 5 是根据本公开实施例的应用管理方法缩容阶段的示意性流程图；

图 6 是根据本公开实施例的应用管理方法扩容阶段的示意性流程图；

图 7 是根据本公开实施例的低功耗内存范围管理的示意图；

图 8 是根据本公开实施例的实例状态转换示意图；

图 9 是系统内核态的内存交换功能的示意图；

图 10 是根据本公开的用户态内存交换功能的示意图；

图 11 示出了根据本公开实施例可用于实现上述应用管理方法的计算设备的结构示意图。

具体实施方式

下面将参照附图更详细地描述本公开的优选实施方式。虽然附图中显示了本公开的优选实施方式，然而应该理解，可以以各种形式实现本公开而不应被这里阐述的实施方式所限制。相反，提供这些实施方式是为了使本公开更加透彻和完整，并且能够将本公开的范围完整地传达给本领域的技术人员。

在本公开的上下文中，术语“第一”、“第二”仅用于区分限定的作用，而没有先后、主次或重要性区分的含义。

为便于描述，在本公开的上下文中，“第一应用实例”用于表示缩容过程中的应用实例。“第二应用实例”用于表示扩容过程中的应用实例。随着应用实例工作状态的切换，第一应用实例和第二应用实例的称谓也可以相互转化。即，第一应用实例经历缩容处理进入低功耗状态后，当要再次经扩容处理时，也可以作为本公开的第二应用实例，反之亦然。

处于低功耗状态的应用实例可以称为“低功耗应用实例”。处于在线状态的应用实例可以称为“线上应用实例”或“在线应用实例”。

在一些场合下，“应用实例”也可以简称为“实例”。

根据本公开，提供了一种无服务器计算（Serverless）场景下基于容器实现的应用管理方案。

无服务器计算弹性是指在无服务器计算模式下，为了应对服务请求流量的变化，应用运行实例需要从一个（或较少的若干个）实例增加到多个（或较多的若干个）实例，或者从多个副本减少为较少的副本，我们把增加和减少的过程分别称为扩容和缩容，扩容和缩容的能力称为弹性。

从是否涉及跨机扩缩容的角度，弹性可以分为垂直弹性和水平弹性，垂直弹性是指发生在同一个主机上的扩缩容行为，水平弹性则是指跨主机的扩缩容行为。

本公开所提到的弹性是指应用程序的扩缩容。应用程序的实例从低功耗（低流量）状态（也可以称为“低功耗运行状态”、“低功耗模式”或“低功耗运行模式”）变为可以接受全量流量的状态（全流量/全功耗状态，也可以称为“线上状态”）为扩容，从接受流量的状态变为待用（Standby）状态为缩容。应当明白，扩容也可以是将实例从低功耗状态转变为部分流量/部分功耗状态，即流量/功耗处于低功耗/低流量状态和全流量/全功耗状态之间的状态。相应地，缩容也可以包括将部分流量/部分功耗状态的实例转变为低功耗/低流量状态。

低功耗模式下，可以限制应用实例运行时的活跃度，减少应用实例资源消耗。

应用实例在低功耗状态下的功耗和/或资源消耗低于在线状态下的功耗和/或资源消耗。

例如，可以设定一个比例值，使得低功耗状态下的功耗和/或资源消耗等于或低于所设定的数值或相应比例值。该比例值例如可以为 10%、5%、2%、1%等等。

另外，也可以针对功耗或不同类型的资源分别设置不同的数值或不同的相应比例值。

在在线状态的应用实例进入低功耗状态时，可以关闭应用实例所使用的部分资源或降低所述部分资源的使用量，而仅保留应用实例所使用的部分系统资源。例如，可以关闭线程池或者降低线程池的线程数量，而保持文件描述符之类的系统资源处于打开状态。

相应地，在低功耗状态的应用实例进入在线状态时，可以恢复所关闭的部分资源或其使用量。

另外，在处于低功耗模式时，应用实例的内存占用空间大幅降低，原来占用的大量内存空间被释放出来，不再使用。这些不再使用的内存空间，包括堆区和非堆区内存空间，都可以清理掉，已分配给其它应用实例。

与直接冻结应用实例不同，在低功耗状态（模式）下，应用实例仍然能够提供一定的服务，例如配置推送等。

图 1 和图 2 示意性地示出了一个应用的缩容和扩容过程中其若干个实例的状态变化。其中作为示例，为一个应用示出了五个实例。应当明白，对应于一个应用的实例个数不

限于此。

另外，图中示出了线上实例（全流量状态）所占用的资源为“4c、4g”，即 4 核（CPU）及 4g 内存，低功耗实例（低功耗状态）所占用的资源为“0.05c、500M”，即 0.05 核（CPU）及 500M 内存。应当明白，资源数量不限于此，而且也不需要每个实例所占用的资源数量相同。而且，各实例所占用的资源还可以根据实时流量来调整。另外，资源的类型也不限于 CPU 和内存，还可以包括其它类型的资源。

图中以实线框表示实例当前状态，以虚线表示先前状态，箭头表示状态变化方向。

图 1 是根据本公开实施例的应用弹性缩容的示意图。

例如响应于针对低流量应用的缩容指令，或例如响应于应用的流量降低到低于第一预定阈值而成为低流量应用，可以执行缩容处理，使一个或多个原来处于线上状态的应用实例进入低功耗状态，转换为低功耗实例。

缩容时，可以根据指令或实时流量，将该应用对应的所有线上实例或其中部分线上实例转换为低功耗实例。

图 2 是根据本公开实施例的应用弹性扩容的示意图。

例如响应于针对低流量应用的扩容指令，或例如响应于低流量应用的流量增加到高于第二预定阈值，可以执行扩容处理，是一个或多个原来处于低功耗状态的应用实例进入在线状态，转换为线上实例。这里，第二预定阈值可以高于或等于第一阈值。

扩容时，也可以根据指令或实时流量，将该应用对应的所有低功耗实例或其中部分低功耗实例转换为线上实例。

下面参考图 3 至图 10 详细描述根据本公开的缩容和扩容处理。

图 3 是根据本公开实施例的应用管理装置的示意性框图。

如图 3 所示，根据本公开实施例的应用管理装置 10 可以包括缩容装置 100、扩容装置 200 和内存交换存储（swap）装置 300。

缩容装置 100 可以包括额度限制装置 110 和配置降低装置 120

扩容装置 200 可以包括额度恢复装置 220 和配置提升装置 210。

图 4 是根据本公开实施例的应用管理方法的示意性流程图。

如图 4 所示，在步骤 S10，配置基于容器实现的无服务器计算系统，以使得允许应用实例在运行时处于在线状态和低功耗状态之一。

通过使得系统支持应用实例运行时（runtime）以低功耗状态/模式工作，系统上层运行时能够了解应用实例是否以低功耗状态工作，可以避免应用程序的行为造成大量的页面输入输出操作（Page In/Page Out），从而避免导致系统不可用。

在缩容阶段，在步骤 S100，例如通过所述缩容装置 100，响应于对应用执行缩容处理，使所述应用的至少一个在线状态的第一应用实例进入低功耗状态

在扩容阶段，在步骤 S200，例如通过所述扩容装置 200，响应于对应用执行扩容处

理，使所述应用的至少一个低功耗状态的第二应用实例进入在线状态。

下面，首先描述根据本公开的缩容处理。如上文所述，这里的缩容处理可以例如响应于针对低流量应用的缩容指令，或例如响应于应用的流量降低到低于第一预定阈值而成为低流量应用，而执行的。

【缩容】

图 5 是根据本公开实施例的应用管理方法缩容阶段步骤 S100 的示意性流程图。

如图 5 所示，在步骤 S110，例如可以通过额度限制装置 110，限制要进行缩容处理的应用的至少一个在线状态的第一应用实例的资源使用额度。

这里，可以基于控制组（Cgroups，Control Groups）机制来限制应用实例的资源使用额度或解除限制。

控制组机制是 Linux 内核提供的一种可以限制单个进程或者一组进程所使用资源的机制。Cgroups 为每种可以控制的资源定义了一个子系统，从而可以对 CPU、内存等资源实现精细化的控制。在云计算环境下，目前流行的容器技术大多深度依赖于 Cgroups 提供的资源限制能力来完成 CPU、内存等部分的资源控制。

相应地，至少一个第一应用实例进入低功耗（运行）状态。

本公开了提出了对传统应用实施低功耗运行的概念。应用在无流量（或低流量）时处于极度不活跃的状态。这个状态下理论上应用需要的 CPU 消耗非常低，内存占用（Memory Footprint）非常小。可以将实例运行的这种状态称为低功耗状态（低功耗运行）。

低功耗运行是应用实例运行的一种状态，应用实例处于存活运行状态，但是仅提供有限的服务。与全功耗（全流量）运行状态相比，低功耗运行状态下占用很少的系统资源，能够快速响应系统外部事件。根据本公开，在特定事件下（例如响应于扩容指令，或响应于低流量应用的流量增加到高于第二预定阈值），运行中的应用实例可以迅速从低功耗运行状态迅速恢复到全功耗运行状态，从而具备全量服务能力，对外承接大量流量请求。

应用实例在运行时进入低功耗运行，上述步骤 S110 中通过 Cgroups 机制，可以将低功耗应用实例的 CPU 和内存规格降低到很低。

一方面，关于 CPU，可以是基于 CPU 共享（CPU share）功能，将多个第一应用实例加入低功耗规格的 CPU 组中运行。该 CPU 组中的应用实例共享 CPU。

由此，可以方便地将分配给低功耗实例的集中到该共享 CPU，进一步降低低功耗运行模式下使用的 CPU 资源，并且还可以将低功耗实例所占用的 CPU 与全功耗实例所使用的 CPU 隔离开来。

另外，在后续执行扩容处理，使第二应用实例进入在线状态时，还可以使第二应用实例退出低功耗规格的 CPU 组。

另一方面，关于内存，下面详细描述根据本公开在低功耗实例管理时的内存管理方

案。

本公开的应用管理方案要求应用程序运行时（Runtime）支持低功耗运行模式。

一些传统Java应用，运行过程中之所以不支持Runtime低功耗运行，是因为JVM（Java虚拟机）内存管理的特征：即使没有外部流量请求，JVM 内存也会在整个堆区内存范围内进行对象分配，且会进行周期性的垃圾回收（GC，Garbage Collection）操作，这些行为都会导致 Java 堆（Heap）中的大量内存区域被触碰（Touch）。GC 操作对象的存活信息更新也会导致大量内存成为脏数据（Dirty）状态。

为了使 Java 运行时（Runtime）支持低功耗运行，首先使 Java 应用程序进入低功耗运行状态。在低功耗状态下，使 Java 进程的垃圾回收（GC）操作只作用于很小的一部分堆区内存范围内。换言之，只进行局部 GC，保证 JVM 的内存占用（Memory Footprint）在低功耗模式下局限在一个范围。

可以利用 AJDK 提供的弹性堆（Elastic Heap）能力，来提供进程低功耗运行能力。在低功耗模式下，GC 可以只作用于限定大小的堆区范围内，Elastic Heap 提供了低功耗运行弹性堆大小设置，通常可以为正常堆范围的 1/10。

另外，在 Java 语言的系统中，不仅可以释放堆区内存，本地（Native）内存部分也可以释放。

上面描述了 Java 语言中的周期性 GC 扫描。其它语言也有各自的低功耗操作，例如释放预定期限内未曾使用过的内存的内存释放操作等，同样也会对处于低功耗状态的应用实例的运行状态造成影响。因此，也可以将这些影响低功耗应用实例的低功耗运行状态的操作限制在一个局部内存范围之内，而将低功耗应用实例占用的内存空间设置在该操作的执行范围之外。

除了这里描述的周期性垃圾回收操作、内存释放操作，其它可能唤起低功耗实例，或影响低功耗实例的低功耗运行状态的周期性内存管理操作的执行范围也都可以加以限制。

可以响应于范围调整指令，调整周期性内存管理操作的执行范围的设定大小，或者调整该执行范围的起止范围。

图 7 是根据本公开实施例的低功耗内存范围管理的示意图。

Java 应用内存 400 中可以包括堆内存（堆区）420 和非堆内存（非堆区）410。

堆内存 420 中，限定周期性内存管理操作的执行范围 421，执行范围 421 之外的范围 422 则为低功耗实例内存范围。在低功耗实例内存范围 422 中，不执行上述周期性内存管理操作。

使应用的至少一个第一应用实例进入低功耗状态时，可以使这些第一应用实例所占用的内存空间处于内存中的低功耗实例内存范围中。

例如，可以将这些第一应用实例占用的内存空间从限定的周期性内存管理操作的执

行范围转移到该执行范围之外的低功耗实例内存范围中。

或者，也可以发出范围调整指令，调整执行范围：在占用内存空间的第一应用实例进入低功耗状态的情况下，设置执行范围不包括所占用内存空间；或者在占用内存空间的第一应用实例进入在线状态的情况下，设置执行范围包括所占用内存空间。

这样，无需实际移动内存数据，即可使应用实例所占用的内存空间，对应于该应用实例的运行状态变化，而在上述执行范围和上述低功耗实例内存范围之间切换。

实例的内存数据除了在线上实例的上述执行范围和低功耗实例内存范围之间切换之外，还可以例如借助操作系统的内存最近最少使用（LRU）算法，将低功耗模式/状态下不活跃的内存进一步换出到交换（Swap）区（内存数据交换设备，或存储设备）。

图 8 是根据本公开实施例的实例状态转换示意图。

如图 8 所示，内存数据的存储逻辑上可以分为三层，即 L0、L1、L2。

L0 层为线上实例的内存数据，位于上述周期性内存管理操作的执行范围之内，上述低功耗实例内存范围之外。

L1 层为低功耗（缓存）实例的内存数据，位于上述低功耗实例内存范围之内，上述周期性内存管理操作的执行范围之外。

L2 层为交换（持久化）到内存交换设备或存储设备的内存数据，例如位于外部硬盘或云盘上。例如可以针对各个容器分别形成镜像，即容器镜像集，实现容器间数据的隔离。

应用实例的内存数据可以随着运行状态的变化而在 L0 层和 L1 层之间切换。同样地，内存数据可以例如基于上述 LRU 算法，在 L1 层和 L2 层之间交换。或者，如下文所述，在扩容时，也可以直接从 L2 层转移到 L1 层。

内存交换（Swap）是当前 Linux 操作系统中内存子系统重要的组成部分。Linux 操作系统使用 Swap 能力来增加节点可用的虚拟内存。借助 Swap 能力，操纵系统可以将主内存中的数据换出持久化到内存交换设备（Swap 设备）上。这样系统可以将所获得（释放出来）的主内存分配给其它的进程（实例）使用。同时，在系统内存不足的情况下，通过 Swap 可以有效地避免系统发生内存溢出（OOM，Out of memory）错误。当这些内存页再次被系统访问的时候，操作系统再从 Swap 设备上读取这些内存页数据，并将其加载到主内存中，从而保证进程的正常运行。

这样，例如可以通过内存交换存储装置 300，使至少一个低功耗应用实例的内存数据在内存和存储设备之间转移。

这里，可以基于最近最少使用算法，选择位于内存的低功耗实例内存范围中的一个或多个低功耗应用实例的内存数据，并将其从内存转移并持久化到存储设备。

响应于换回指令或流量请求或实例部署策略变化，将位于存储设备上的一个或多个低功耗应用实例的内存数据转移回到内存的低功耗实例内存范围中。可以使用高速网盘

作为用于存储所换出内存的 Swap 设备（内存数据交换设备/存储设备）。

另外，可以使用操作系统默认的 Swap 能力/策略，即，系统内核态（Kernel）的内存交换功能，来执行内存数据交换。

但 Linux 系统目前的 Swap 能力，物理主机共享相同的 Swap 存储空间，多个应用实例换出（从内存换出到存储设备）的内存数据会存储到多个主机（Host）共享的存储设备上。因此，Swap 并不支持 Cgroup 隔离。

图 9 是系统内核态的内存交换功能的示意图。

应用 A 的低功耗实例 A1、A2；应用 B 的低功耗实例 B1、B2 分别具有相应的内存数据。为便于理解，图中以不同的内部阴影填充方式表示不同实例的数据。

如图 9 所示，所有低功耗实例的内存数据都交换存储到同一个内存交换存储设备上，即共享相同的内存数据交换设备/存储设备。例如在频繁换入换出时，内存交换设备中的序列会被打乱，随机化太严重，严重影响读写性能。

因此本公开也允许通过配置使用一种用户态（Userspace）的 Swap 能力，即，用户态的内存交换功能。通过用户态的内存交换功能，能够实现容器、进程、实例层面的 Swap 隔离。例外，可以采用格式化的内存结构，实现大力度地预读，实现快上快下。

用户态的内存交换功能，可以采用文件映射、顺序读写方式，读写性能能够适应不同存储介质的顺序读写性能。这样，一定程度上也可以降低对高速随机读 Swap 存储设备能力的依赖。

图 10 是根据本公开的用户态内存交换功能的示意图。

通过用户态的设置，可以将对应于不同实例或来自不同容器的内存数据交换存储到不同的存储设备，或存储设备的不同页面上，实现实例或容器数据的隔离。

这样，可以使用系统内核态的内存交换功能，使一个或多个低功耗应用实例的内存数据在内存和存储设备之间转移。也可以使用用户态的内存交换功能，使一个或多个低功耗应用实例的内存数据在内存和存储设备之间转移，并且将来自不同容器的内存数据转移到不同的存储设备，或存储设备的不同页面。

针对应用或应用实例，可以由相关工作人员来配置使用内核态内存交互功能还是用户态内存交互功能。

可以接收例如相关工作人员发出的针对应用或应用实例的内存交换设置指令。

内存交换设置指令可以用于指示设置针对该应用的应用实例或该应用实例，使用系统内核态的内存交换功能还是用户态的内存交换功能来执行内存数据转移。

这样，响应于所接收到的内存交换设置指令，可以针对该应用的应用实例或该应用实例，设置用于执行内存数据转移的内存交换功能。

在使用用户态的内存交换功能的情况下，可以使用多个超性能云盘设备构造具有相同优先级的内存交换用存储设备。

通过如上所述针对 CPU 和内存进行低功耗转换处理之后，目标应用实例已进入低功耗运行状态。不活跃的内存数据被进一步已经换出到低成本的外部 IO 设备（内存数据交换设备/存储设备）上。

此时，可以是进一步对应用实例的资源进行降配，把实例低功耗运行节省的资源归还给调度系统，以用于调度其它高优先级的任务，实现资源的复用。

返回图 5，在步骤 S120，例如可以通过配置降低装置 120，降低至少一个第一应用实例各自所在容器的资源配置。

容器释放的资源，会通过原地升级容器荚（Pod）层面资源规格的方式还给集群，然后集群可以把其它的工作负载（Workload）的容器荚（Pod）调度到该 Node 上。

这里，例如可以基于如 Kubernetes（K8s）的容器资源原地升级机制，来降低或提升容器的资源配置。在降低容器的资源配置时，基于容器荚资源原地升级机制，将应用实例所在容器所释放的资源归还给调度系统。相应地，在扩容阶段提升容器的资源配置时，基于容器荚资源原地升级机制，从调度系统为容器申请资源。

这里，资源管理包含两个层面：容器内部运行时层面和 K8s 层面。例如，原来系统给应用实例分配了 4C8G（4 核 8G 内存）。运行时用了这么多资源，而进入低功耗状态后，应用实例自己却只占用了 1/10 的资源。但是如果不在 K8s 层面也对这个资源进行更新的话，调度层会仍然认为应用实例占用了 4C8G 的资源。这样便不能将空余出来的资源分配给其它应用实例或容器。这里，K8s 层面主要是调度层面维护的元数据。

容器资源原地升级能够在不影响 Kubernetes 容器荚（Pod）的情况下，实现 Pod 资源的原地更新，降低当前应用实例所在容器的资源配置。从而可以综合实现应用实例低功耗运行和容器平台资源的配额调整。

容器荚（Pod）是 Kubernetes 中能够创建和部署的最小单元，是 Kubernetes 集群中的一个应用实例，总是部署在同一个节点（Node）上。容器荚（Pod）中包含一个或多个容器，还包括了存储、网络等各容器共享的资源。容器荚（Pod）可以支持多种容器环境，例如当前流行的容器环境 Docker 等。

至此，已详细描述了根据本公开的缩容处理，即限制低流量应用的至少一个第一应用实例的资源使用额度，使其进入低功耗（运行）状态；并进一步降低这些第一应用实例各自所在容器的资源配置，将从上述容器释放的资源分配给该容器所在容器荚（Pod）中的其它容器。

【扩容】

下面，描述根据本公开的扩容处理。如上文所述，这里的扩容处理可以是例如响应于针对低流量应用的扩容指令，或例如响应于低流量应用的流量增加到高于第二预定阈值，而执行的。

在应用缩容之后，其至少一个应用实例处于低功耗运行模式。低功耗运行模式能够

支持动态资源配置更新。

例如，当有流量请求，或流量增大，需要扩容的时候，可以将当前处于低功耗运行第二应用实例快速拉起，以具备承接全部流量的能力。

图 6 是根据本公开实施例的应用管理方法扩容阶段步骤 S200 的示意性流程图。

如图 6 所示，在步骤 S210，例如可以通过配置提升装置 210，提升要扩容的应用的低功耗状态的至少一个第二应用实例各自所在容器的资源配置。

如上文所述相同，例如，可以基于容器资源原地升级机制，来提升容器荚内容器的资源配置。

这里，可以优先选择使容器所在节点中资源相对空闲的机器上的容器中的第二应用实例进入在线状态。

即，扩容会优先选择容器所在节点中资源比较空闲的机器上的第二应用实例，向调度系统申请资源升配到原来的规格或设定的规格或基于当前流量确定的规格。

换言之，在要扩容的应用具有多个处于低功耗状态的应用实例，不同应用实例处于不同机器上的情况下，优先将资源相对空闲的机器上的应用实例恢复到在线状态，并相应恢复其资源使用额度，提升其所在容器的资源配置。

容器资源配置升配完成之后，可以解除本地 Cgroup 限制，使应用实例运行时退出低功耗运行模式（低功耗状态），即进入在线模式（在线状态）。

即在步骤 S220，例如可以通过额度恢复装置 220，解除至少一个第二应用实例的资源使用额度限制。

与上文所述相同，这里也可以基于控制组机制来解除对应用实例的资源使用额度的限制。

另外，还可以基于用户态的检查点恢复（CRIU）技术的实时迁移功能，将低功耗应用实例迁移到相对空闲的机器上，从而可以进一步解决资源挤兑的问题。

这样，低功耗运行模式需要解决扩容时，针对扩容实例所在机器资源不足的问题，可以引入了基于用户态的检查点恢复（Checkpoint/Restore）（CRIU，Checkpoint/Restore in Userspace）技术的实时迁移（Live Migration，也可以称为“热迁移”或“在线迁移”）功能，将低功耗运行实例迁移到空闲的目标机器上进行调度。即，可以对低功耗应用实例进行水平迁移。

一方面，可以在应用实例将要进入在线状态而其所在节点上资源不足的情况下，基于用户态的检查点恢复（CRIU）技术的实时迁移功能，将所述应用实例迁移到相对空闲的节点上。

另一方面，在同一个节点上具有多个低功耗状态的应用实例的情况下，可以例如预判将来这些低功耗应用实例要恢复为在线应用实例时，该节点上资源将不足，即可能出现资源挤兑问题。这样，可以基于用户态的检查点恢复（CRIU）技术，将一个或多个所

述应用实例做成检查点快照，然后基于所述检查点快照在相对空闲的节点上恢复所述一个或多个应用实例这里，例如可以使至少一个第二应用实例所占用的内存空间处于内存中的上述低功耗实例内存范围之外，也即上述周期性内存管理操作在内存中的执行范围之内。

当第二应用实例的内存数据已经被转移到内存交换设备/存储设备/外部 IO 设备时，例如可以通过内存交换存储装置 300，快速换入存储在外部 IO 设备上的内存数据，即，将第二应用实例的内存数据从存储设备转移到内存中的执行范围之中。

另外，默认的系统内核态内存交换功能的 Swap 换入（从存储设备换入到内存）属于懒人（Lazy）模式，不适合快速弹起场景。

为了适应无服务器计算（Serverless）极速换入的要求，可以在用户态内存交换功能中实现一层并发换入逻辑。

可以使用用户态的内存交换功能，同时将多个低功耗应用实例的内存数据从内存分别转移到不同的存储设备或存储设备的不同页面。另一方面，也可以使用用户态的内存交换功能，同时将多个低功耗应用实例的内存数据从不同的存储设备或存储设备的不同页面转移到内存中。

这样，能够使内存交换一次性换入速度达到 IO 上限，大幅提升扩容性能。

并且，为了加速换入时的性能，提升换入时输入输出吞吐量（IO Throughput）上限，在底层内存交换设备/存储设备构造上，可以是利用 Swap 子系统的优先级特性，用多个 ESSD（Enhanced SSD，超性能云盘）块设备来构造同一个优先级内存交换设备/存储设备，实现内存交换的 RAID（独立冗余磁盘阵列）能力。在实际使用过程中，能够使内存数据换入的 IO 吞吐量大幅提升，实现极速的秒级弹性。

例如，在一个应用示例中，将单个 1.2T 的 ESSD 设备替换为 3 个 400G 的 ESSD 设备，IO 吞吐量可以从 350M/s 提升到 960M/s。

Serverless 强调应用实例的快速创建。本公开的方案可以利用 Go 语言的协程特性，结合 Linux pagemap 映射结构，能够对多段内存数据同时访问。由此，提供了一种在 Serverless 场景下，实现内存快速换入的方法。与传统操作系统基于 Lazy 访问加载的方式相比，能够充分利用 IO 最大带宽以将换出到外部存储介质的内存数据加载到内存中，快速提供服务。

这里，详细描述了根据本公开的扩容处理，即提升第二应用实例各自所在容器的资源配置；解除第二应用实例的资源使用额度限制，使第二应用实例退出低功耗运行模式/低功耗状态。

至此，已详细描述了根据本公开的基于容器的应用管理方案中的缩容和扩容处理。

由此，本公开提出了一种应用管理方案，其可以用于实现存量应用的快速弹性，而无需对应用架构做任何改变。

本公开在云计算 Serverless 场景下，提出应用实例的低功耗运行模式。

结合应用实例的低功耗运行模式，结合操作系统内核态的 Swap 能力和用户态 Swap 能力，以及例如 K8s 层面的容器原地升级能力（容器归还资源过程），提供了一套基于用户态来控制应用实例低功耗运行的方法。

可以使低功耗状态下缓存的存活对象占用极少的系统资源。而当流量请求到达需要扩容的阈值时，可以在秒级时间内水平扩容线上服务应用实例，并启动时申请资源。

另外，在实施例中，还通过在用户态进行并发控制，并利用 Swap 优先级特性提供了一套高性能的内存换出/换入方案，可以实现 Serverless 场景下实现秒级弹性扩容。

另外，还可以对应用的弹性缩容、扩容的一些相关信息进行记录、展示或统计，以便进行评估、调整，必要时可以进行干预。例如，相关信息可以包括低功耗应用实例和在线应用实例各自的数量、两种状态的应用实例之间的数量比例、相应状态持续时间、应用实例从低功耗状态进入在线状态所耗用时间、应用扩容所耗用时间、应用缩容所耗用时间等等。

根据本公开的应用管理方法的应用缩容、扩容方案可以适用于多种应用场景，特别是流量随时间变化较大的应用场景，换言之，一些在特定时间段需要同时应对巨大的流量，而另外一些时间所需要应对的流量则相对小很多，需要弹性缩容或扩容的场景。例如，特定时段进行的大型促销活动，又例如火车售票等场景。在这样的应用场景中，会在极短的时间段内突然爆发出数倍甚至数十倍、数百倍于平时的流量。而本公开的应用管理方案能够很好地应对这些场景下应用缩容、扩容的需要。

图 11 示出了根据本公开实施例可用于实现上述应用管理方法的计算设备的结构示意图。

参见图 11，计算设备 1000 包括存储器 1010 和处理器 1020。

处理器 1020 可以是一个多核的处理器，也可以包含多个处理器。在一些实施例中，处理器 1020 可以包含一个通用的主处理器以及一个或多个特殊的协处理器，例如图形处理器（GPU）、数字信号处理器（DSP）等等。在一些实施例中，处理器 1020 可以使用定制的电路实现，例如特定用途集成电路（ASIC，Application Specific Integrated Circuit）或者现场可编程逻辑门阵列（FPGA，Field Programmable Gate Arrays）。

存储器 1010 可以包括各种类型的存储单元，例如系统内存、只读存储器（ROM），和永久存储装置。其中，ROM 可以存储处理器 1020 或者计算机的其他模块需要的静态数据或者指令。永久存储装置可以是可读写的存储装置。永久存储装置可以是即使计算机断电后也不会失去存储的指令和数据的非易失性存储设备。在一些实施方式中，永久性存储装置采用大容量存储装置（例如磁或光盘、闪存）作为永久存储装置。另外一些实施方式中，永久性存储装置可以是可移除的存储设备（例如软盘、光驱）。系统内存可以是可读写存储设备或者易失性可读写存储设备，例如动态随机访问内存。系统内存

可以存储一些或者所有处理器在运行时需要的指令和数据。此外，存储器 1010 可以包括任意计算机可读存储媒介的组合，包括各种类型的半导体存储芯片（DRAM，SRAM，SDRAM，闪存，可编程只读存储器），磁盘和/或光盘也可以采用。在一些实施方式中，存储器 1010 可以包括可读和/或写的可移除的存储设备，例如激光唱片（CD）、只读数字多功能光盘（例如 DVD-ROM，双层 DVD-ROM）、只读蓝光光盘、超密度光盘、闪存卡（例如 SD 卡、min SD 卡、Micro-SD 卡等等）、磁性软盘等等。计算机可读存储媒介不包含载波和通过无线或有线传输的瞬间电子信号。

存储器 1010 上存储有可执行代码，当可执行代码被处理器 1020 处理时，可以使处理器 1020 执行上文述及的应用管理方法。

上文中已经参考附图详细描述了根据本发明的应用管理方案。

此外，根据本发明的方法还可以实现为一种计算机程序或计算机程序产品，该计算机程序或计算机程序产品包括用于执行本发明的上述方法中限定的上述各步骤的计算机程序代码指令。

或者，本发明还可以实施为一种非暂时性机器可读存储介质（或计算机可读存储介质、或机器可读存储介质），其上存储有可执行代码（或计算机程序、或计算机指令代码），当所述可执行代码（或计算机程序、或计算机指令代码）被电子设备（或计算设备、服务器等）的处理器执行时，使所述处理器执行根据本发明的上述方法的各个步骤。

本领域技术人员还将明白的是，结合这里的公开所描述的各种示例性逻辑块、模块、电路和算法步骤可以被实现为电子硬件、计算机软件或两者的组合。

附图中的流程图和框图显示了根据本发明的多个实施例的系统和方法的可能实现的体系架构、功能和操作。在这点上，流程图或框图中的每个方框可以代表一个模块、程序段或代码的一部分，所述模块、程序段或代码的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。也应当注意，在有些作为替换的实现中，方框中所标记的功能也可以以不同于附图中所标记的顺序发生。例如，两个连续的方框实际上可以基本并行地执行，它们有时也可以按相反的顺序执行，这依所涉及的功能而定。也要注意的是，框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合，可以用执行规定的功能或操作的专用的基于硬件的系统来实现，或者可以用专用硬件与计算机指令的组合来实现。

以上已经描述了本发明的各实施例，上述说明是示例性的，并非穷尽性的，并且也不限于所披露的各实施例。在不偏离所说明的各实施例的范围和精神的情况下，对于本技术领域的普通技术人员来说许多修改和变更都是显而易见的。本文中所用术语的选择，旨在最好地解释各实施例的原理、实际应用或对市场中的技术的改进，或者使本技术领域的其它普通技术人员能理解本文披露的各实施例。

权 利 要 求 书

1．一种基于容器的应用管理方法，包括：

配置基于容器实现的无服务器计算系统，以使得允许应用实例在运行时处于在线状态和低功耗状态之一，其中，应用实例在低功耗状态下的功耗和/或资源消耗低于在线状态下的功耗和/或资源消耗；

响应于对应用执行缩容处理，使所述应用的至少一个在线状态的第一应用实例进入低功耗状态；以及

响应于对应用执行扩容处理，使所述应用的至少一个低功耗状态的第二应用实例进入在线状态。

2．根据权利要求 1 所述的方法，其中，

使所述应用的至少一个在线状态的第一应用实例进入低功耗状态的步骤包括：

限制所述至少一个第一应用实例的资源使用额度；以及

降低所述至少一个第一应用实例各自所在容器的资源配置，

并且/或者，使所述应用的至少一个低功耗状态的第二应用实例进入在线状态的步骤包括：

提升所述至少一个第二应用实例各自所在容器的资源配置；以及

解除所述至少一个第二应用实例的资源使用额度限制。

3．根据权利要求 2 所述的方法，其中，使所述应用的至少一个低功耗状态的第二应用实例进入在线状态的步骤包括：

优先选择使容器所在节点中资源相对空闲的机器上的容器中的第二应用实例进入在线状态。

4．根据权利要求 2 所述的方法，还包括：

在应用实例将要进入在线状态而其所在节点上资源不足的情况下，基于用户态的检查点恢复 CRIU 技术的实时迁移功能，将所述应用实例迁移到相对空闲的节点上；以及/或者

在同一个节点上具有多个低功耗状态的应用实例的情况下，基于用户态的检查点恢复（CRIU）技术，将一个或多个所述应用实例做成检查点快照，然后基于所述检查点快照在相对空闲的节点上恢复所述一个或多个应用实例。

5．根据权利要求 2 所述的方法，其中，

基于控制组机制来限制应用实例的资源使用额度或解除所述限制；并且/或者

在降低容器的资源配置时，基于容器荚资源原地升级机制，将所述应用实例所在容器所释放的资源归还给调度系统；以及/或者

在提升容器的资源配置时，基于容器荚资源原地升级机制，从调度系统为所述容器申请资源。

6．根据权利要求 1 所述的方法，其中，

使所述至少一个第一应用实例进入低功耗状态的步骤包括：基于 CPU 共享功能，将多个所述第一应用实例加入低功耗规格的 CPU 组中运行，所述低功耗规格的 CPU 组中的应用实例共享 CPU；并且/或者

使所述至少一个第二应用实例进入在线状态的步骤包括：使第二应用实例退出所述低功耗规格的 CPU 组。

7．根据权利要求 1 所述的方法，其中，

使所述至少一个第一应用实例进入低功耗状态的步骤包括：使所述至少一个第一应用实例所占用的内存空间处于内存中的低功耗实例内存范围中；并且/或者

使所述至少一个第二应用实例进入在线状态的步骤包括：使所述至少一个第二应用实例所占用的内存空间处于内存中的低功耗实例内存范围之外。

8．根据权利要求 7 所述的方法，其中，所述低功耗实例内存范围是内存中周期性内存管理操作在内存中的执行范围之外的范围。

9．根据权利要求 8 所述的方法，其中，所述周期性内存管理操作包括内存垃圾回收操作和/或释放预定期限内未曾使用过的内存的内存释放操作。

10．根据权利要求 8 所述的方法，还包括：

响应于范围调整指令，调整所述执行范围和/或所述执行范围的大小；以及/或者

在占用内存空间的第一应用实例进入低功耗状态的情况下，设置所述执行范围不包括所述第一应用实例所占用的内存空间；以及/或者

在占用内存空间的第一应用实例进入在线状态的情况下，设置所述执行范围包括所述第一应用实例所占用的内存空间。

11．根据权利要求 1 所述的方法，其中，使所述至少一个第一应用实例进入低功耗状态的步骤包括：

关闭第一应用实例所使用的部分资源或降低所述部分资源的使用量，仅保留第一应用实例所使用的部分系统资源。

12．根据权利要求 1 所述的方法，还包括：

使一个或多个低功耗状态的应用实例的内存数据在内存和存储设备之间转移。

13．根据权利要求 12 所述的方法，其中，使一个或多个低功耗应用实例的内存数据在内存和存储设备之间转移的步骤包括：

使用系统内核态的内存交换功能，使所述一个或多个低功耗应用实例的内存数据在内存和存储设备之间转移；以及/或者

使用用户态的内存交换功能，使所述一个或多个低功耗应用实例的内存数据在内存和存储设备之间转移，并且将来自不同容器的内存数据转移到不同的存储设备或存储设备的不同页面。

14．根据权利要求 13 所述的方法，其中，使用用户态的内存交换功能，同时将多个低功耗应用实例的内存数据从内存分别转移到不同的存储设备或存储设备的不同页

面，并且/或者，同时将多个低功耗应用实例的内存数据从不同的存储设备或存储设备的不同页面转移到内存中。

15．根据权利要求 13 所述的方法，还包括：

接收针对应用或应用实例的内存交换设置指令，所述内存交换设置指令用于指示设置针对该应用的应用实例或该应用实例，使用系统内核态的内存交换功能还是用户态的内存交换功能来执行内存数据转移；

响应于所述内存交换设置指令，针对该应用的应用实例或该应用实例，设置用于执行内存数据转移的内存交换功能。

16．根据权利要求 13 所述的方法，其中，在使用用户态的内存交换功能的情况下，使用多个超性能云盘设备构造具有相同优先级的内存交换用存储设备。

17．根据权利要求 12 所述的方法，其中，所述一个或多个低功耗状态的应用实例的内存数据在内存和存储设备之间转移的步骤包括：

基于最近最少使用算法，选择一个或多个低功耗状态的应用实例的内存数据，并将其从内存转移并持久化到存储设备；以及/或者

响应于换回指令或流量请求或实例部署策略变化，将位于存储设备上的一个或多个低功耗状态的应用实例的内存数据转移回到内存中。

18．根据权利要求 12 所述的方法，其中，在所述第二应用实例的内存数据已被转移到存储设备上的情况下，使所述应用的至少一个低功耗状态的第二应用实例进入在线状态的步骤还包括：

将所述第二应用实例的内存数据从存储设备转移到内存中周期性内存管理操作的执行范围之中。

19．根据权利要求 1 所述的方法，其中，

响应于针对应用的缩容指令，或响应于应用的流量降低到低于第一预定阈值而成为低流量应用，对所述应用执行缩容处理；并且/或者

响应于针对应用的扩容指令，或响应于应用的流量增加到高于第二预定阈值，对所述应用执行扩容处理。

20．一种基于容器的应用管理装置，部署于基于容器实现的无服务器计算系统中，所述无服务器计算系统和被配置为允许应用实例在运行时处于在线状态和低功耗状态之一，其中，应用实例在低功耗状态下的功耗和/或资源消耗低于在线状态下的功耗和/或资源消耗，该装置包括：

缩容装置，响应于对应用执行缩容处理，使所述应用的至少一个在线状态的第一应用实例进入低功耗状态；以及

扩容装置，响应于对应用执行扩容处理，使所述应用的至少一个低功耗状态的第二应用实例进入在线状态。

21．根据权利要求 20 所述的装置，其中，所述缩容装置包括：

额度限制装置，用于限制低流量应用的至少一个第一应用实例的资源使用额度；以及

配置降低装置，用于降低所述至少一个第一应用实例各自所在容器的资源配置。

22．根据权利要求 20 所述的装置，其中，所述扩容装置包括：

配置提升装置，用于提升所述至少一个第一应用实例中至少一个第二应用实例各自所在容器的资源配置；以及

额度恢复装置，用于解除所述至少一个第二应用实例的资源使用额度限制。

23．根据权利要求 20 至 22 中任何一项所述的装置，还包括：

内存交换存储装置，使所述至少一个第一应用实例中的一个或多个第一应用实例的内存数据在内存和存储设备之间转移。

24．一种计算设备，包括：

处理器；以及

存储器，其上存储有可执行代码，当所述可执行代码被所述处理器执行时，使所述处理器执行如权利要求 1 至 19 中任何一项所述的方法。

25．一种非暂时性机器可读存储介质，其上存储有可执行代码，当所述可执行代码被电子设备的处理器执行时，使所述处理器执行如权利要求 1 至 19 中任何一项所述的方法。