(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 114265704 B
(45)授权公告日 2022.05.17

(21)申请号 202210200963.0

(22)申请日 2022.03.03

(65)同一申请的已公布的文献号
申请公布号 CN 114265704 A

(43)申请公布日 2022.04.01

(73)专利权人 环球数科集团有限公司
地址 518063 广东省深圳市南山区粤海街道高新南九道10号深圳湾科技生态园10栋B座17层01-03号

(72)发明人 张卫平 丁烨 米小武 李显阔

(74)专利代理机构 北京清控智云知识产权代理事务所(特殊普通合伙) 11919
专利代理师 马肃

(51)Int.Cl.
G06F 9/50(2006.01)
G06F 9/48(2006.01)
G06F 9/455(2006.01)

(56)对比文件
CN 111190688 A,2020.05.22
CN 106506244 A,2017.03.15
CN 109639791 A,2019.04.16
CN 109714400 A,2019.05.03
WO 2021095943 A1,2021.05.20
刘哲源等.基于模拟退火算法的粒子群优化算法在容器调度中的应用.《计算机测量与控制》.2021,第29卷(第12期),第177-183页.

审查员 杨林靖

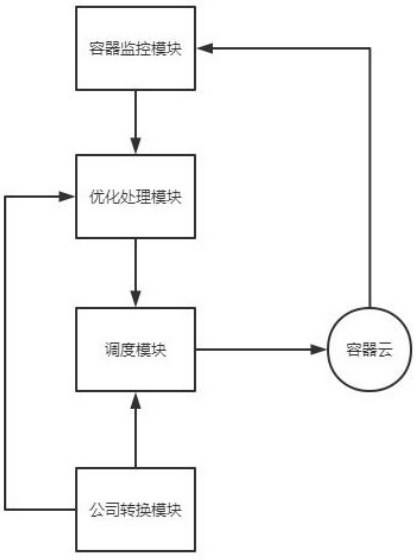
权利要求书2页 说明书7页 附图4页

(54)发明名称

一种基于凸优化的混合容器云调度系统

(57)摘要

本发明提供了一种基于凸优化的混合容器云调度系统,包括容器监控模块、优化处理模块、调度模块和公私转换模块,所述容器监控模块用于监控容器在公有云中的运行状态,所述优化处理模块基于公有云中容器的运行状态处理得到优化方案,所述调度模块根据所述优化方案将容器在公有云中进行调动,所述公私转换模块用于处理用户提交的容器转换申请,并通过所述调度模块将容器在公有云和私有云中进行调动。本系统能够对公有云中的容器进行动态调度,保障公有云的稳定性,在调度过程中,采用了凸优化的方式,能够更快地找到合适的调度方案。



CN 114265704 B

1.一种基于凸优化的混合容器云调度系统,其特征在于,包括容器监控模块、优化处理模块、调度模块和公私转换模块,所述容器监控模块用于监控容器在公有云中的运行状态,所述优化处理模块基于公有云中容器的运行状态处理得到优化方案,所述调度模块根据所述优化方案将容器在公有云中进行调动,所述公私转换模块用于处理用户提交的容器转换申请,并通过所述调度模块将容器在公有云和私有云中进行调动;

所述优化处理模块在公有云中选择出需要进行容器调度的第一目标节点和接收调度容器的第二目标节点,并基于凸优化的方式将调度容器中的应用程序在所述第二目标节点中进行分配,使得公有云处于稳定的运行状态;

所述优化处理模块在选择所述第一目标节点时通过计算节点的待调指数Adi:

$$Adi=\frac{(1-Sri)}{\sum_{j=1}^{n}(1-Srj)}\bullet\frac{n(n+1)}{2};$$

其中,n为公有云中节点的数量,Sri为第i个节点的运行状态值,Srj为第j个节点的运行状态值;

当所述待调指数小于阈值时,对应的节点为第一目标节点;

所述优化处理模块在选择第二目标节点并分配应用程序时,建立目标函数$R(U)$:

$$R(U)=\sum_{k=1}^{m}\left(\frac{(x_k-\sum x)}{Zx_k}\bullet+\frac{(y_k-\sum y)}{Zy_k}\right)\bullet Sri;$$

其中,m为待选第二目标节点的数量,Zx_k表示第k个待选第二目标节点的CPU总量值,Zy_k表示第k个待选第二目标节点的内存总量值,x_k表示第k个待选第二目标节点的剩余CPU总量值,y_k表示第k个待选第二目标节点的剩余内存总量值,$\sum x$表示该待选第二目标节点分配到的所有应用程序的CPU使用量值,$\sum y$表示该待选第二目标节点分配到的所有应用程序的内存使用量值,U为应用程序与待选第二目标节点的匹配向量;

所述优化处理模块建立约束函数集:

$$\begin{cases}x_k>\sum x;\\ y_k>\sum y;\end{cases}$$

所述优化处理模块在满足约束函数集条件的前提下计算得到使目标函数$R(U)$最大的匹配向量U,匹配向量中分配到应用程序的待选第二目标节点为正式的第二目标节点,所述调度模块基于匹配向量U进行调度;

其中,公有云中节点的运行状态值的计算公式为:

$$Sr=(1-Ur(CPU))\bullet(1-Ur(RM));$$

其中,$Ur(CPU)$表示该节点中的CPU使用率,$Ur(RM)$表示该节点中的内存使用率;

所述优化处理模块通过如下步骤来选择第一目标节点中需要调度的容器或容器中的应用程序;

S1、将所述第一目标节点中以容器为单位统计其CPU的使用率和内存的使用率，并用$Urt(CPU)$表示第t个容器的该节点中的CPU使用率，用$Urt(RM)$表示第t个容器的内存使用率；

S2、所述优化处理模块计算第t个容器的调度优先度Prt：

$$Prt=\frac{Urt(CPU)\bullet Ur^2(CPU)+Urt(RM)\bullet Ur^2(RM)}{Urt(CPU)+Urt(RM)};$$

S3、选择调度优先度最大的容器；

S4、判断该容器是否需要整体调度，判断标准如下：

$$\begin{cases}Urt(CPU)<\alpha\bullet Ur(CPU)\\Urt(RM)<\alpha\bullet Ur(RM)\end{cases};$$

其中，α为比例系数；

当同时满足上述两个不等式时，将对应的容器整体调度并结束该过程，否则跳至步骤S5；

S5、将步骤S3中选择的容器中的每个应用程序计算其使用综合值Zh：

$$Zh=App\_Ur(CPU)+App\_Ur(RM);$$

其中，$APP_Ur(CPU)$表示对应的应用程序的CPU使用率，$APP_Ur(RM)$表示对应的应用程序的内存使用率；

S6、将所有应用程序根据综合值Zh从高到低排序，并设置一个结果池用于放置需要调度的应用程序；

S7、按序获取应用程序的使用数据，根据下述进行判断：

$$\begin{cases}APP\_Ur(CPU)+Rs\_Ur(CPU)<\alpha\bullet Ur(CPU)\\APP\_Ur(RM)+Rs\_Ur(RM)<\alpha\bullet Ur(RM)\end{cases};$$

其中，$Rs_Ur(CPU)$表示结果池中所有应用程序的CPU使用率总和，$Rs_Ur(RM)$表示结果池中所有应用程序的内存使用率总和；

当同时满足这两个不等式时，将该应用程序放入结果池中；

S8、重复步骤S7，直至所有应用程序判断完毕；

S9、将结果池中的所有应用程序打包作为需要调度的整体。

2.如权利要求1所述的一种基于凸优化的混合容器云调度系统，其特征在于，公有云中的一个节点中含有多个容器，每个容器内运行有多个应用程序，所述优化处理模块在所述第一目标节点中选出一个容器或该容器中的部分应用程序进行调度。

3.如权利要求2所述的一种基于凸优化的混合容器云调度系统，其特征在于，所述公私转换模块接收的调度申请分为两类，一类是从公有云向私有云进行调度，在通过审核后由所述调度模块直接进行调度，另一类是私有云向公有云进行调度，在通过审核后，需要所述优化处理模块计算处理得到匹配向量U，再由所述调度模块进行调度。

一种基于凸优化的混合容器云调度系统

技术领域

[0001] 本公开涉及容器云领域，具体涉及一种基于凸优化的混合容器云调度系统。

背景技术

[0002] 容器是一种相比于虚拟机更轻量级，更灵活的虚拟化处理方式，它将一个应用程序所需的一切打包在一起，容器包括所有代码，各种依赖甚至操作系统，这让应用程序几乎在任何地方都可以运行，因此它的诞生，解决了一个重要问题：如何确保应用程序从一个环境移动到另一个环境的正确运行，在云中运行的容器称为容器云，由于容器云设计了大量的数据，为了保证容器云的稳定运行，需要一种调度系统将容器在不同的云节点中进行调度，

[0003] 现在已经开发出了很多容器调度系统，经过我们大量的检索与参考，发现现有的授权系统有如公开号为KR101784681B1，KR101807806B1、CN106506655B和KR100443450B1所公开的系统，通过创建一个容器镜像，用于创建容器组；容器组是以同一个镜像创建的容器集合；然后按照调度、拓展策略，由调度器进行调度、拓展。但该系统在具体进行调度的时候无法快速有效的找到最佳的调度方案，从而影响了调度效率。

发明内容

[0004] 本发明的目的在于，针对所存在的不足，提出了一种基于凸优化的混合容器云调度系统，

[0005] 本发明采用如下技术方案：

[0006] 一种基于凸优化的混合容器云调度系统，包括容器监控模块、优化处理模块、调度模块和公私转换模块，所述容器监控模块用于监控容器在公有云中的运行状态，所述优化处理模块基于公有云中容器的运行状态处理得到优化方案，所述调度模块根据所述优化方案将容器在公有云中进行调动，所述公私转换模块用于处理用户提交的容器转换申请，并通过所述调度模块将容器在公有云和私有云中进行调动；

[0007] 所述优化处理模块在公有云中选择出需要进行容器调度的第一目标节点和接收调度容器的第二目标节点，并基于凸优化的方式将调度容器中的应用程序在所述第二目标节点中进行分配，使得公有云处于稳定的运行状态；

[0008] 所述优化处理模块在选择所述第一目标节点时通过计算节点的待调指数Adi：

[0009] $$Adi=\frac{(1-Sri)}{\sum_{j=1}^{n}(1-Srj)}\bullet\frac{n(n+1)}{2}$$；

[0010] 其中，n为公有云中节点的数量，Sri为第i个节点的运行状态值；

[0011] 当所述待调指数小于阈值时，对应的节点为第一目标节点；

[0012] 所述优化处理模块在选择第二目标节点并分配应用程序时，建立目标函数$R(U)$：

[0013] $$R(U)=\sum_{i=1}^{m}\left(\frac{(x_i-\sum x)}{Zx_i}\bullet+\frac{(y_i-\sum y)}{Zy_i}\right)\bullet Sri;$$

[0014] 其中，m为待选第二目标节点的数量，Zx_i表示第i个待选第二目标节点的CPU总量值，Zy_i表示第i个待选第二目标节点的内存总量值，x_i表示第i个待选第二目标节点的剩余CPU总量值，y_i表示第i个待选第二目标节点的剩余内存总量值，$\sum x$表示该待选第二目标节点分配到的所有应用程序的CPU使用量值，$\sum y$表示该待选第二目标节点分配到的所有应用程序的内存使用量值，U为应用程序与待选第二目标节点的匹配向量；

[0015] 所述优化处理模块建立约束函数集：

[0016] $$\begin{cases}x_i>\sum x;\\ y_i>\sum y;\end{cases}$$

[0017] 所述优化处理模块在满足约束函数集条件的前提下计算得到使目标函数$R(U)$最大的匹配向量U，匹配向量中分配到应用程序的待选第二目标节点节点为正式的第二目标节点，所述调度模块基于匹配向量U进行调度；

[0018] 进一步的，公有云中节点的运行状态值的计算公式为：

[0019] $$Sr=(1-Ur(CPU))\bullet(1-Ur(RM));$$

[0020] 其中，$Ur(CPU)$表示该节点中的CPU使用率，$Ur(RM)$表示该节点中的内存使用率；

[0021] 进一步的，公有云中的一个节点中含有多个容器，每个容器内运行有多个应用程序，所述优化处理模块在所述第一目标节点中选出一个容器或该容器中的部分应用程序进行调度；

[0022] 进一步的，所述优化处理模块计算出第一目标节点中第i个容器的调度优先度Pri：

[0023] $$Pri=\frac{Uri(CPU)\bullet Ur^2(CPU)+Uri(RM)\bullet Ur^2(RM)}{Uri(CPU)+Uri(RM)};$$

[0024] 其中，$Uri(CPU)$为第i个容器在该第一目标节点中的CPU使用率，$Uri(RM)$为第i个容器在该第一目标节点中的内存使用率；

[0025] 所述优化处理模块选择调度优先度最大的容器作为调度对象；

[0026] 进一步的，所述公私转换模块接收的调度申请分为两类，一类是从公有云向私有云进行调度，在通过审核后由所述调度模块直接进行调度，另一类是私有云向公有云进行调度，在通过审核后，需要所述优化处理模块计算处理得到匹配向量U，再由所述调度模块进行调度。

[0027] 本发明所取得的有益效果是：

[0028] 本系统通过实时监控公有云中各节点容器运行的状态，自动地对容器进行调度，使整个公有云中的容器能够稳定地运行，在调度过程中，通过资源使用率来选择被调度的

节点，通过资源使用量来选择接收调度容器的节点，并依据凸优化来分配调度容器中的应用程序，使得能够更快更有效地找到分配方案执行，提高了调度效率，本系统还结合了公有云和私有云，能够根据用户的需求在公有云与私有云之间进行数据调度。

[0029] 为使能更进一步了解本发明的特征及技术内容，请参阅以下有关本发明的详细说明与附图，然而所提供的附图仅用于提供参考与说明，并非用来对本发明加以限制。

附图说明

[0030] 图1为本发明整体结构框架示意图；

[0031] 图2为本发明公有云中节点、容器和应用程序的关系示意图；

[0032] 图3为本发明结果池中放入应用程序的示意图；

[0033] 图4为本发明第一目标节点中选择需调度的容器或容器内的应用程序的流程示意图；

[0034] 图5为本发明公私转换的两种模式示意图。

具体实施方式

[0035] 以下是通过特定的具体实施例来说明本发明的实施方式，本领域技术人员可由本说明书所公开的内容了解本发明的优点与效果。本发明可通过其他不同的具体实施例加以施行或应用，本说明书中的各项细节也可基于不同观点与应用，在不悖离本发明的精神下进行各种修饰与变更。另外，本发明的附图仅为简单示意说明，并非依实际尺寸的描绘，事先声明。以下的实施方式将进一步详细说明本发明的相关技术内容，但所公开的内容并非用以限制本发明的保护范围。

[0036] 实施例一。

[0037] 本实施例提供了一种基于凸优化的混合容器云调度系统，结合图1，包括容器监控模块、优化处理模块、调度模块和公私转换模块，所述容器监控模块用于监控容器在公有云中的运行状态，所述优化处理模块基于公有云中容器的运行状态处理得到优化方案，所述调度模块根据所述优化方案将容器在公有云中进行调动，所述公私转换模块用于处理用户提交的容器转换申请，并通过所述调度模块将容器在公有云和私有云中进行调动；

[0038] 所述优化处理模块在公有云中选择出需要进行容器调度的第一目标节点和接收调度容器的第二目标节点，并基于凸优化的方式将调度容器中的应用程序在所述第二目标节点中进行分配，使得公有云处于稳定的运行状态；

[0039] 所述优化处理模块在选择所述第一目标节点时通过计算节点的待调指数Adi：

[0040] $$Adi=\frac{(1-Sri)}{\sum_{j=1}^{n}(1-Srj)}\bullet\frac{n(n+1)}{2}$$；

[0041] 其中，n为公有云中节点的数量，Sri为第i个节点的运行状态值；

[0042] 当所述待调指数小于阈值时，对应的节点为第一目标节点；

[0043] 所述优化处理模块在选择第二目标节点并分配应用程序时，建立目标函数$R(U)$：

[0044] $$R(U)=\sum_{i=1}^{m}\left(\frac{(x_i-\sum x)}{Zx_i}\bullet+\frac{(y_i-\sum y)}{Zy_i}\right)\bullet Sri;$$

[0045] 其中，m为待选第二目标节点的数量，Zx_i表示第i个待选第二目标节点的CPU总量值，Zy_i表示第i个待选第二目标节点的内存总量值，x_i表示第i个待选第二目标节点的剩余CPU总量值，y_i表示第i个待选第二目标节点的剩余内存总量值，$\sum x$表示该待选第二目标节点分配到的所有应用程序的CPU使用量值，$\sum y$表示该待选第二目标节点分配到的所有应用程序的内存使用量值，U为应用程序与待选第二目标节点的匹配向量；

[0046] 所述优化处理模块建立约束函数集：

[0047] $$\begin{cases}x_i>\sum x;\\ y_i>\sum y;\end{cases}$$

[0048] 所述优化处理模块在满足约束函数集条件的前提下计算得到使目标函数$R(U)$最大的匹配向量U，匹配向量中分配到应用程序的待选第二目标节点节点为正式的第二目标节点，所述调度模块基于匹配向量U进行调度；

[0049] 公有云中节点的运行状态值的计算公式为：

[0050] $$Sr=(1-Ur(CPU))\bullet(1-Ur(RM));$$

[0051] 其中，$Ur(CPU)$表示该节点中的CPU使用率，$Ur(RM)$表示该节点中的内存使用率；

[0052] 公有云中的一个节点中含有多个容器，每个容器内运行有多个应用程序，所述优化处理模块在所述第一目标节点中选出一个容器或该容器中的部分应用程序进行调度；

[0053] 所述优化处理模块计算出第一目标节点中第i个容器的调度优先度Pri：

[0054] $$Pri=\frac{Uri(CPU)\bullet Ur^2(CPU)+Uri(RM)\bullet Ur^2(RM)}{Uri(CPU)+Uri(RM)};$$

[0055] 其中，$Uri(CPU)$为第i个容器在该第一目标节点中的CPU使用率，$Uri(RM)$为第i个容器在该第一目标节点中的内存使用率；

[0056] 所述优化处理模块选择调度优先度最大的容器作为调度对象；

[0057] 所述公私转换模块接收的调度申请分为两类，一类是从公有云向私有云进行调度，在通过审核后由所述调度模块直接进行调度，另一类是私有云向公有云进行调度，在通过审核后，需要所述优化处理模块计算处理得到匹配向量U，再由所述调度模块进行调度。

[0058] 实施例二。

[0059] 本实施例包含了实施例一的全部内容，提供了一种基于凸优化的混合容器云调度系统，包括容器监控模块、优化处理模块、调度模块和公私转换模块，所述容器监控模块用于监控容器在公有云中的运行状态，所述优化处理模块基于公有云中容器的运行状态处理得到优化方案，所述调度模块根据所述优化方案将容器在公有云中进行调动，所述公私转换模块用于处理用户提交的容器转换申请，并通过所述调度模块将容器在公有云和私有云

中进行调动；

[0060] 结合图2，在公有云中设有多个节点，每个节点上装有多个容器，不同的容器能够提供不同的运行环境，每个容器上运行有多个应用程序，所述容器监控模块将采集每个应用程序在运行时所使用的CPU和内存资源数据，并将每个容器上使用的CPU和内存资源数据进行汇总，并将汇总数据发送给所述优化处理模块；

[0061] 所述优化处理模块接收多个节点上发送的汇总数据后进行优化分析；

[0062] 所述优化处理模块计算每个节点的运行状态值Sr：

[0063] $Sr=(1-Ur(CPU))\bullet(1-Ur(RM))$；

[0064] 其中，$Ur(CPU)$表示该节点中的CPU使用率，$Ur(RM)$表示该节点中的内存使用率；

[0065] 所述优化处理模块将所有节点根据其运行状态值从高到低排序，并用Sri表示第i个节点的运行状态值，然后通过下式计算整个公有云中第i个节点的待调指数Adi：

[0066] $$Adi=\frac{(1-Sri)}{\sum_{j=1}^{n}(1-Srj)}\bullet\frac{n(n+1)}{2}$$；

[0067] 其中，n为公有云中节点的数量；

[0068] 当所述待调指数小于阈值时，对应节点中的容器或者容器中的应用程序需进行调度，将这些节点称为第一目标节点；

[0069] 由上述公式的性质可知，当某个节点不需要调度，则排在其后面的节点也无需进行调度；

[0070] 结合图3和图4，所述优化处理模块通过如下步骤来选择第一目标节点中需要调度的容器或容器中的应用程序；

[0071] S1、将所述第一目标节点中以容器为单位统计其CPU的使用率和内存的使用率，并用$Uri(CPU)$表示第i个容器的该节点中的CPU使用率，用$Uri(RM)$表示第i个容器的内存使用率；

[0072] S2、所述优化处理模块计算第i个容器的调度优先度Pri：

[0073] $$Pri=\frac{Uri(CPU)\bullet Ur^2(CPU)+Uri(RM)\bullet Ur^2(RM)}{Uri(CPU)+Uri(RM)}$$；

[0074] S3、选择调度优先度最大的容器；

[0075] S4、判断该容器是否需要整体调度，判断标准如下：

[0076] $$\begin{cases}Uri(CPU)<\alpha\bullet Ur(CPU)\\Uri(RM)<\alpha\bullet Ur(RM)\end{cases}$$；

[0077] 其中，α为比例系数；

[0078] 当同时满足上述两个不等式时，将对应的容器整体调度并结束该过程，否则跳至步骤S5；

[0079] S5、将步骤S3中选择的容器中的每个应用程序计算其使用综合值Zh：

[0080] $Zh=App\_Ur(CPU)+App\_Ur(RM)$；

[0081] 其中，$APP_Ur(CPU)$表示对应的应用程序的CPU使用率，$APP_Ur(RM)$表示对应的应用程序的内存使用率；

[0082] S6、将所有应用程序根据综合值Zh从高到低排序，并设置一个结果池用于放置需要调度的应用程序；

[0083] S7、按序获取应用程序的使用数据，根据下述进行判断：

[0084] $$\begin{cases} APP\_Ur(CPU)+Rs\_Ur(CPU)<\alpha\bullet Ur(CPU) \\ APP\_Ur(RM)+Rs\_Ur(RM)<\alpha\bullet Ur(RM) \end{cases};$$

[0085] 其中，$Rs_Ur(CPU)$表示结果池中所有应用程序的CPU使用率总和，$Rs_Ur(RM)$表示结果池中所有应用程序的内存使用率总和；

[0086] 当同时满足这两个不等式时，将该应用程序放入结果池中；

[0087] S8、重复步骤S7，直至所有应用程序判断完毕；

[0088] S9、将结果池中的所有应用程序打包作为需要调度的整体；

[0089] 所述优化处理模块在非第一目标节点中的所有节点中选择出合适的节点用于接收需要调度的容器或应用程序包，这些节点称为第二目标节点；

[0090] 所述优化处理模块在选择第二目标节点时采用凸优化的方式进行选取，将需调度的容器或应用程序包中的每个应用程序分配给其中一个所述第二目标节点，所述优化处理模块建立目标函数$R(U)$：

[0091] $$R(U)=\sum_{i=1}^{m}\left(\frac{(x_i-\sum x)}{Zx_i}\bullet+\frac{(y_i-\sum y)}{Zy_i}\right)\bullet Sri;$$

[0092] 其中，m为待选第二目标节点的数量，Zx_i表示第i个待选第二目标节点的CPU总量值，Zy_i表示第i个待选第二目标节点的内存总量值，x_i表示第i个待选第二目标节点的剩余CPU总量值，y_i表示第i个待选第二目标节点的剩余内存总量值，$\sum x$表示该待选第二目标节点分配到的所有应用程序的CPU使用量值，$\sum y$表示该待选第二目标节点分配到的所有应用程序的内存使用量值，U为应用程序与待选第二目标节点的匹配向量；

[0093] 所述优化处理模块建立约束函数集：

[0094] $$\begin{cases} x_i>\sum x; \\ y_i>\sum y; \end{cases}$$

[0095] 所述优化处理模块在满足约束函数集条件的前提下计算得到使目标函数$R(U)$最大的匹配向量U，匹配向量中分到到应用程序的待选第二目标节点节点称为正式的第二目标节点，所述优化处理模块将匹配向量U发送至所述调度模块，所述调度模块依据匹配向量对需要调度的容器或应用程序包中的应用程序进行调度；

[0096] 结合图5，所述公私转换模块接收用户发送的调度申请，当所述调度申请是从公有云中调度数据至私有云时，在通过对用户的审核后，所述调度模块直接依据调度申请进行调度，当所述调度申请是从私有云调度数据至公有云时，在通过对用户的审核后，所述优化

处理模块先根据需调度数据的CPU使用量和内存使用量选择合适的第二目标节点，所述调度模块再执行调度。

[0097] 以上所公开的内容仅为本发明的优选可行实施例，并非因此局限本发明的保护范围，所以凡是运用本发明说明书及附图内容所做的等效技术变化，均包含于本发明的保护范围内，此外，随着技术发展其中的元素可以更新的。

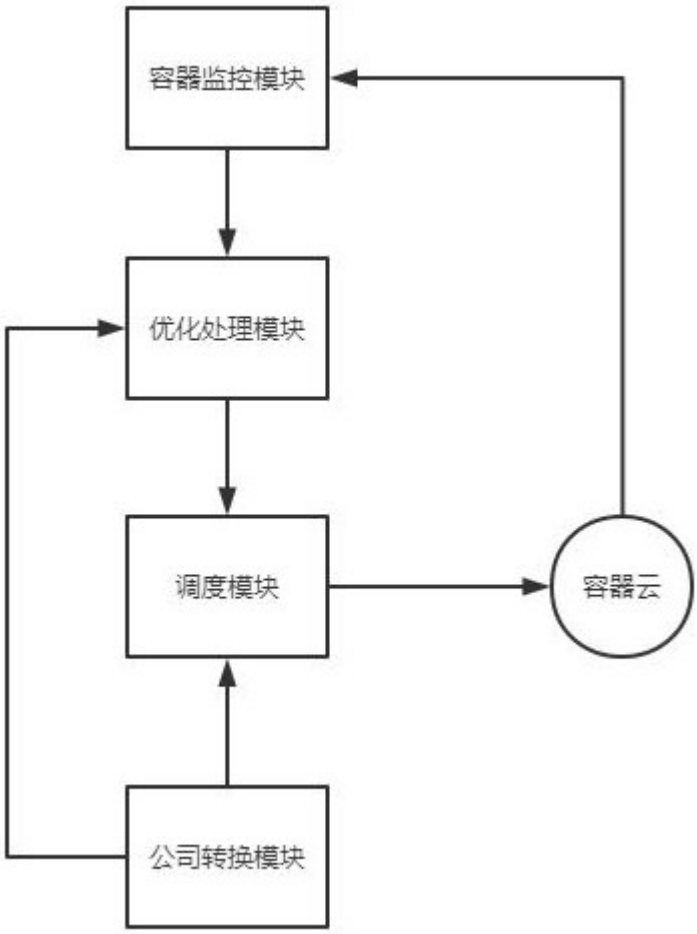


图1

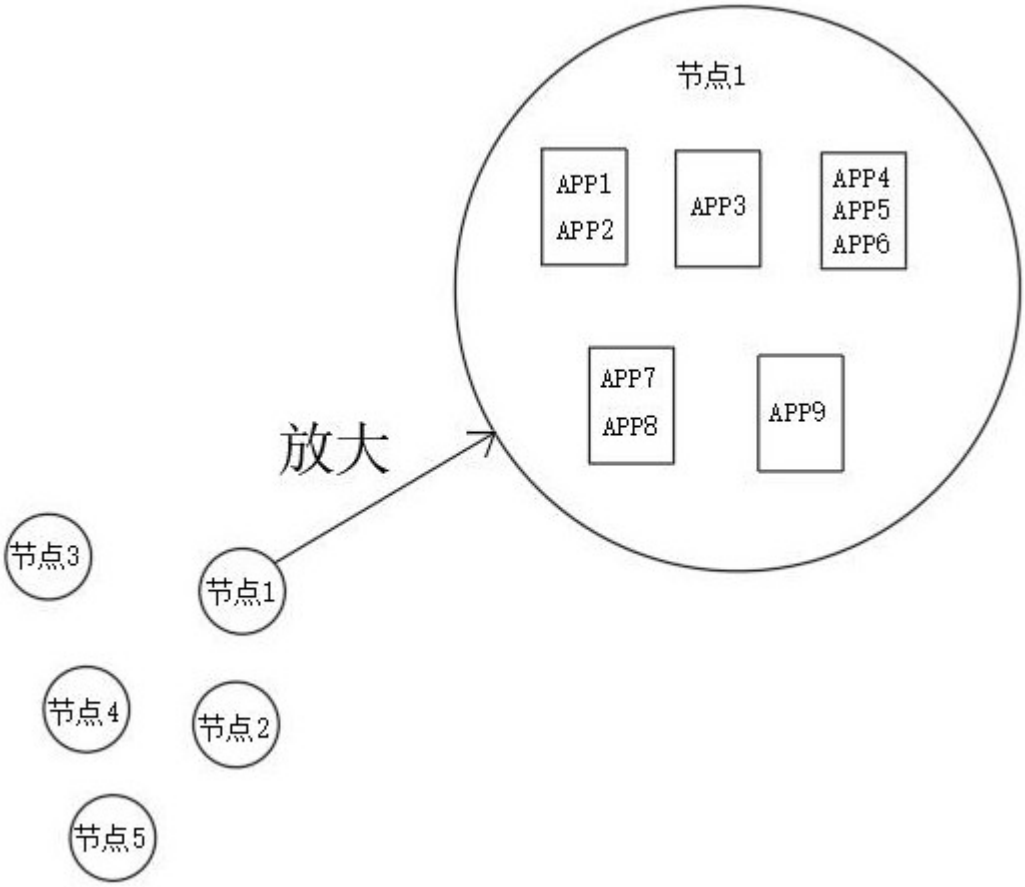

节点1
APP1
APP2
APP3
APP4
APP5
APP6
APP7
APP8
APP9
放大
节点3
节点1
节点4
节点2
节点5

图2

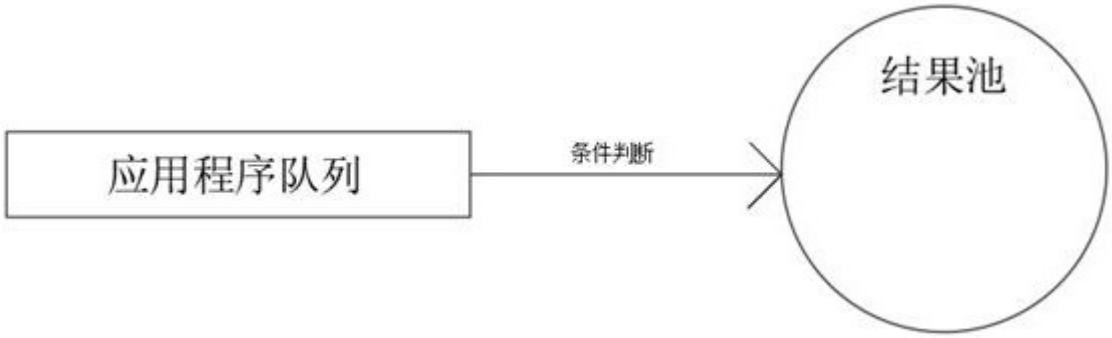

结果池
应用程序队列
条件判断

图3

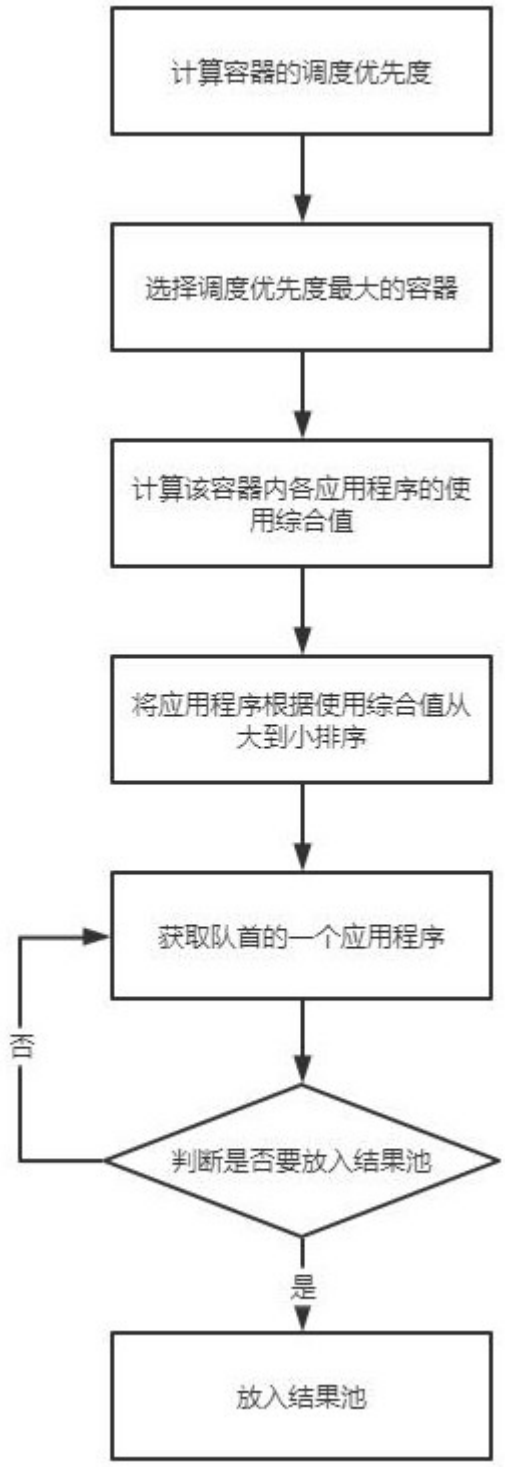

计算容器的调度优先度
选择调度优先度最大的容器
计算该容器内各应用程序的使用综合值
将应用程序根据使用综合值从大到小排序
获取队首的一个应用程序
否
判断是否要放入结果池
是
放入结果池

图4

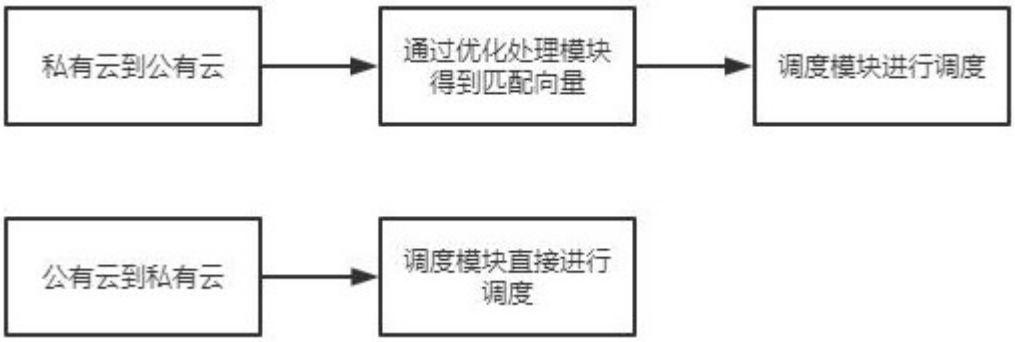


图5