ISSN 1004-9037,CODEN SCYCE4
Journal of Data Acquisition and Processing Vol. 30,No. 3,May 2015,pp. 519－527
DOI:10. 16337/j. 1004-9037. 2015. 03. 006

http://sjcj. nuaa. edu. cn
E-mail:sjcj@nuaa. edu. cn
Tel/Fax:＋86-025-84892742

云数据中心网络中的先应式碎片整理算法

郭　磊　侯维刚

(东北大学信息科学与工程学院,沈阳,110819)

摘　要:在云数据中心网络内,虚拟机(Virtual machine, VM)被动态创建和下线,这就导致资源碎片不被后续 VM 请求所利用。为解决上述问题,以最小化使用服务器数量为目标的服务器整合技术被提出。虽然这种方法可以在某一时间段内减少资源碎片,但却付出了较大的 VM 迁移代价。因此本文提出了一种基于预测的先应式碎片整理算法,在减少无效 VM 迁移的同时,将资源碎片重新整合为可用的连续资源,从而最大化 VM 收益。文中对此问题进行了数学定义,随后设计了启发式方法获取近似最优解。仿真结果表明,所提算法能够获取最大收益,并能够大幅度降低 VM 迁移成本。

关键词:云数据中心网络;虚拟机迁移;资源碎片整理

中图分类号:TN711. 1　　**文献标志码**:A

Provident Resource Defragmentation Algorithm for Cloud Data Center Network

Guo Lei, Hou Weigang

(College of Information Science and Engineering, Northeastern University, Shenyang, 110819, China)

Abstract: In a cloud data center network, virtual machines (VMs) can be dynamically created and terminated, causing the fact that fragmented resources cannot be further utilized. To solve the problem, the server consolidation is proposed to minimize the number of active servers. Although the approach can reduce resource fragmentation at particular time, it may be over aggressive at the price of too frequent VM migration. Therefore, a novel provident resource defragmentation algorithm is proposed to reduce the unnecessary VM migration and maxmize the profit by re-consolidating fragments into available continuous resources. An optimization problem is formulated, and a heuristic method for obtaining near-optimal results is developed. Extensive numerical results confirm that the framework provides the highest profit and significantly reduces the VM migration cost.

Key words: cloud data center network; virtual machine migration; resource defragmentation

引　　言

近年来,Amazon EC2, Microsoft Azure, Infrastructure-asa-service(IaaS)等云数据中心平台快速发展,并实现了多个应用服务提供商对底层物理基础设施的共享[1]。由于虚拟机(Virtual machine,VM)的动态生成、重组与下线,云数据中心网络资源易产生碎片,从而导致较低的资源利用率[2,3]。图 1(a)

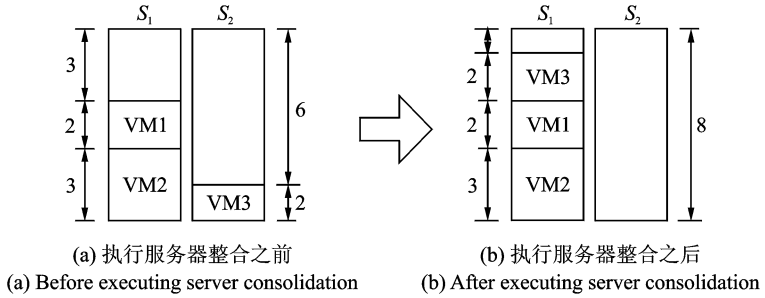
基金项目:国家自然科学基金(61401082,61471109)资助项目;中央高校基本科研业务费专项资金(N130817002,N130404002)资助项目;辽宁省教育厅(L2014089)资助项目。
收稿日期:2015-03-15;**修订日期**:2015-04-12

为产生上述资源碎片的现象举例,其中服务器 S_1 和 S_2 承载了 3 个 VM,且各自的总容量为 8 个单位资源。VM1 和 VM3 各占用 2 个单位资源,而 VM2 占用 3 个单位资源。因此在图 1(a)所示的网络状态下,若一个占用 7 个单位资源的 VM 请求到达,则无法被此网络容纳,从而产生资源碎片。

现有云数据中心网络资源碎片整理方案可归纳为两大类:基于 VM 请求到达的迁移策略,以及服务器整合策略。在第一种策略中,当一个 VM 请求到达后,云数据中心网络首先判断自己能否容纳这个 VM[4,5]。若不能,它将对已存放在服务器内的一个或多个 VM 进行迁移处理,使其中的一个服务器能够腾出空间来容纳到达的 VM[6]。这是个可行策略,但是,当上百个 VM 需要在短时间内置入服务器中时,云数据中心网络将执行频繁的 VM 迁移,这会导致严重的服务中断和延迟。

第二种策略是服务器整合技术,它通过减少服务器的使用数量以腾出更多空闲服务器来容纳后续到达的 VM[7-15]。如图 1(b)所示,将 VM3 迁入服务器 S_1 后可腾空 S_2,从而容纳即将到达的、占用 7 个单位资源的 VM 请求。可见,此技术可在一定程度上减少资源碎片,但腾空服务器会产生频繁且无效的 VM 迁移,系统灵活性也较差。仍以图 1(a)为例,若即将到达的两个 VM 请求分别占用 5 个和 3 个单位资源,则无需采用服务器整合,便可直接将它们放入网络中。

因此,一个更加高效的云数据中心网络资源碎片整理方案应重点关注两个方面的问题:(1)何时执行必要的碎片整理操作;(2)在碎片整理进程中,哪些 VM 必须进行迁移。针对上述两个关注点,本文提出了云数据中心网络中基于预测的先应式碎片整理算法。首先,本文对上述问题进行数学定义,在给定网络中现有 VM 集合的条件下,对后续即将到达的 VM 集合进行预测,从而求出最大限度避免无效 VM 迁移的碎片整理方案(即最优解)。由于此问题是 NP 困难的,本文还设计了有效的启发式方法获取近似最优解。



(a) 执行服务器整合之前
(a) Before executing server consolidation

(b) 执行服务器整合之后
(b) After executing server consolidation

图 1 资源碎片与服务器整合技术示意图

Fig. 1 Diagram of resource fragments and server consolidation

1 问题定义

给定一个具有 S 个服务器的云数据中心网络。其中,对于服务器 $j(1\leqslant j\leqslant S)$:$O_j$ 为总容量;$L_j(L_j\leqslant O_j)$为占用资源;F_j 为可用剩余资源,且 $F_j=O_j-L_j$。根据惯例(如 Amazon EC2),假设云数据中心网络将受理 C 种类型 VM。每个 VM 请求为一个三元组$\langle c,e_c,r_c\rangle$。其中,c 为类型索引号;e_c 为付费;r_c 为资源需求。为便于讨论,假设 r_c 为一实数,且 $r_1>r_2>\cdots>r_C$。资源需求越大,付费越高,因此有 $e_1>e_2>\cdots>e_C$。在特定时间点 t,云数据中心网络内可存在先前放置的 VM(即现有 VM)、一些新到达或离开此网络的 VM。此外,可对特定未来时间点 $t'(t'>t)$的 VM 到达与离开状态做出预测,从而在时间点 t 求解有效的 VM 迁移决策。一些重要的符号定义如下(以首个字母表排序):

$\boldsymbol{A}$ 为二维 VM 放置矩阵;C 为类型总数;e_c 为容纳一个类型-c 的 VM 所获得的收入;F_j 为第 j 个服务器的可用剩余资源;G_s 为存放迁出 VM 的服务器集合;G_r 为具有可用剩余资源的服务器集合;i 为 VM 索引号;j 为服务器索引号;l_r 为迁移成本系数$(0<l_r<1)$;l_u 为服务器成本系数$(0<l_u<1)$;L_j 为第 j 个服务器的占用资源;m_c 为网络内现有类型-c 的 VM 数;n_c 为在时间点 t 到达的、类型-c 的 VM 数;n'

为将在时间点 t' 到达的 VM 总数;n'_c 为将在时间点 t' 到达的、类型-c 的 VM 数;O_j 为第 j 个服务器的总容量;$p_c(n'_c)$ 为将在时间点 t' 到达的 n'_c 个类型-c 的 VM 的概率;ϕ_c 为将在时间点 t' 到达的类型-c 的 VM 比例,且 $\Phi=\{\phi_c\}_{c\in[1,C]}$;$r_c$ 为类型-c 的 VM 的资源需求;S 为网络内服务器总数;t 为当前时间点;t' 为未来时间点;U_j 为一个布尔变量,用以记录第 j 个服务器是否空闲。

1.1　VM 迁移的最佳放置

考虑 VM 迁移的最佳放置问题为:在时间点 t,采用 $m_c\times S$ 矩阵 $\boldsymbol{A}^{t^-}(c)$ 记录网络内现有类型-c 的 VM 放置情况:矩阵元素 $a(c)_{ij}=1$ 表示第 i 个类型-c 的 VM 放置在第 j 个服务器内;否则 $a(c)_{ij}=0$。采用 $n_c\times S$ 矩阵 $\boldsymbol{A}^{t,\beta}(c)$ 记录在时间点 t 新到达的、类型-c 的 VM 放置情况:矩阵元素 $a(c)^{\beta}_{ij}$ 的取值原则与 $a(c)_{ij}$ 类似。为容纳更多新到达的 VM 请求,需迁移网络内部分现有 VM。为此,采用 $m_c\times S$ 矩阵 $\boldsymbol{A}^{t,\alpha}(c)$ 记录迁移后网络内现有的类型-c 的 VM 放置情况:其元素 $a(c)^{\alpha}_{ij}$ 的取值原则与 $a(c)_{ij}$ 类似。相应地,计算总收入如下

$$\pi=\sum_{c=1}^{C}e_c\cdot\sum_{i=1}^{n_c}\sum_{j=1}^{S}[a(c)^{\beta}_{ij}] \tag{1}$$

计算总迁移和服务器成本如下

$$\kappa=\sum_{c=1}^{C}\sum_{i=1}^{m_c}\sum_{j=1}^{S}\{l_r\cdot r_c\cdot a(c)_{ij}\cdot[1-a(c)^{\alpha}_{ij}]+l_u\cdot U_j\} \tag{2}$$

其中,$\forall\, j\in[1,S]$

$$U_j=\begin{cases}0 & \sum_{i=1}^{m_c}a(c)^{\alpha}_{ij}+\sum_{i=1}^{n_c}a(c)^{\beta}_{ij}=0\\ 1 & \text{其他}\end{cases} \tag{3}$$

对现有 VM 的放置应遵循以下约束

$$\sum_{j=1}^{S}a(c)_{ij}=1 \qquad \forall\, i\in[1,m_c],\forall\, c\in[1,C] \tag{4}$$

$$\sum_{c=1}^{C}r_c\cdot\Big[\sum_{i=1}^{m_c}a(c)_{ij}\Big]\leqslant O_j \qquad \forall\, j\in[1,S] \tag{5}$$

新到达的 VM 的放置以及对现有 VM 的迁移应遵循以下约束

$$\sum_{j=1}^{S}a(c)^{\alpha}_{ij}=1 \quad \forall\, i\in[1,m_c],\forall\, c\in[1,C] \tag{6}$$

$$\sum_{j=1}^{S}a(c)^{\beta}_{ij}\leqslant 1 \quad \forall\, i\in[1,n_c],\forall\, c\in[1,C] \tag{7}$$

$$\sum_{c=1}^{C}r_c\cdot\Big[\sum_{i=1}^{m_c}a(c)^{\alpha}_{ij}+\sum_{i=1}^{n_c}a(c)^{\beta}_{ij}\Big]\leqslant O_j \qquad \forall\, j\in[1,S] \tag{8}$$

式(4,6)确保 VM 不被分割;式(7)表明采用迁移后,某些 VM 请求仍可能被阻塞;式(5,8)表明可容纳的 VM 数受限于服务器的总容量。则考虑 VM 迁移的最佳放置问题可描述为

$$\text{Maximize}(\pi-\kappa) \tag{9}$$

1.2　基于预测的先应式碎片整理

基于预测的先应式碎片整理问题为:假设在时间点 t 无 VM 到达,且没有 VM 在未来时间点 t' 之前离开网络。给定 $p_c(n'_c)$ 为将在时间点 t' 到达的、n'_c 个类型-c 的 VM 的概率。采用 $n'_c\times S$ 矩阵 $\boldsymbol{A}^{t',\beta}(c)$ 记录将在时间点 t' 到达的、类型-c 的 VM 放置情况:矩阵元素 $a(c)^{\beta'}_{ij}=1$ 表示第 i 个类型-c 的 VM 将放置在第 j 个服务器内;否则 $a(c)^{\beta'}_{ij}=0$。相应地,推导出可预测的总收入如下

$$\pi'=\sum_{c=1}^{C}e_c\cdot\sum_{n'_c=0}^{\infty}p_c(n'_c)\cdot\sum_{i=1}^{n'_c}\sum_{j=1}^{S}[a(c)^{\beta'}_{ij}] \tag{10}$$

由于迁移成本只与现有 VM 有关,仍可根据式(2)计算总迁移和服务器成本。类似地,未来到达 VM 的放置及对现有 VM 的迁移遵循以下约束

$$\sum_{j=1}^{S} a(c)_{ij}^{\beta'} \leqslant 1 \qquad \forall i \in [1, n'_c], \forall c \in [1, C] \tag{11}$$

$$\sum_{c=1}^{C} r_c \cdot \left[\sum_{i=1}^{m_c} a(c)_{ij}^{\alpha} + \sum_{i=1}^{n'_c} a(c)_{ij}^{\beta'}\right] \leqslant O_j \quad \forall j \in [1, S] \tag{12}$$

则基于预测的先应式碎片整理问题描述如下

$$\text{Maximize}(\pi' - \kappa) \tag{13}$$

实际上,任一时间点新到达 VM 请求数均是一个有限值。因此,假定可得到未来时间点 t' 新到达的 VM 数预测值为 n'。给定向量 $\boldsymbol{\Phi}=\{\phi_c\}_{c\in[1,C]}$(其中 ϕ_c 为在未来时间点 t' 到达的、类型-c 的 VM 比例),从而有

$$\sum_{c=1}^{C} \phi_c = 1 \tag{14}$$

$$n'_c = n' \times \phi_c \tag{15}$$

相应地,遵循式(6,11,12)约束,将可预测的总收入化简为

$$\overline{\pi}' = \sum_{c=1}^{C} e_c \cdot \sum_{i=1}^{n'_c} \sum_{j=1}^{S} [a(c)_{ij}^{\beta'}] \tag{16}$$

则基于预测的先应式碎片整理问题可描述为

$$\text{Maximize}(\overline{\pi}' - \kappa) \tag{17}$$

2 算法描述

在时间点 t,本文所提出的基于预测的先应式碎片整理(Provident resource defragmentation, PRD)算法主流程描述如下:

输入:$A^{t-}=\{A^{t-}(c)\}_{c\in[1,C]}$, $\Phi=\{\phi_c\}_{c\in[1,C]}$, n'

输出:$A^{t,\alpha}$, $A^{t',\beta'}$

(1)$A^{t,\alpha}(0,0)=A^{t-}$; $(A^{t',\beta'}(0,0), n^t) \leftarrow \text{BVF}(A^{t-}, \Phi, n')$

(2)If 所有 n' 个 VM 均已成功放置 then

Return$(A^{t,\alpha}(0,0), A^{t',\beta'}(0,0))$

(3)for $c=1,2,\cdots,C$ do

If $n_c^t < n' \cdot \phi_c$ then

$(A^{t,\alpha}(c,k_c^*), A^{t',\beta'}(c,k_c^*)) \leftarrow$

MaxPD$(A^{t,\alpha}(c-1,k_{c-1}^*), A^{t',\beta'}(c-1,k_{c-1}^*), n'_c, n_c^t)$

else

$k_c^* = n_c^t$; $A^{t,\alpha}(c,k_c^*) = A^{t,\alpha}(c-1,k_{c-1}^*)$

$A^{t',\beta'}(c,k_c^*) = A^{t',\beta'}(c-1,k_{c-1}^*)$

end

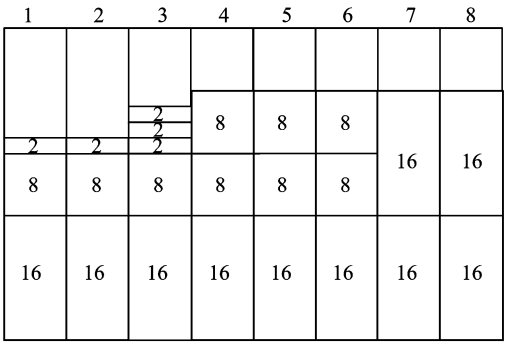
资源需求较大优先(Bigger volume first, BVF)子算法:从具有最大资源需求的 VM 开始,依次尝试将在未来时间点 t' 到达的 VM 放入服务器中。随后得到临时分配矩阵 $\boldsymbol{A}^{t',\beta'}(0,0)$ 以及向量 $\boldsymbol{n}^t=\{n_c^t\}_{c\in[1,C]}$,其中 n_c^t 为采用 BVF 方法成功放入服务器中的、类型-c 的 VM 数,从而得到相应收入和成本如下

$$\overline{\pi}_{0,0} = \sum_{c=1}^{C} e_c \cdot n_c^t \tag{18}$$

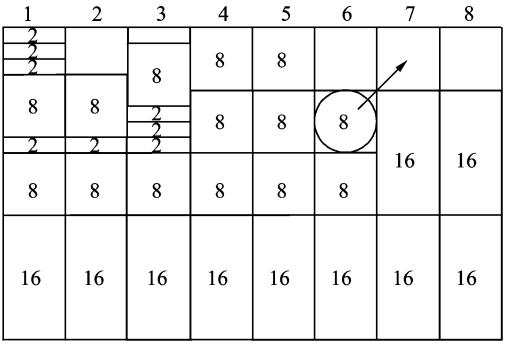
$$\kappa_{0,0} = 0 \tag{19}$$

首先,初始化 $n'\times S$ 空矩阵 $\boldsymbol{A}^{t',\beta}(0,0)$。随后以类型索引号的升序对 VM 进行初始放置:对于某种类型 VM,遍历所有服务器以获取能够容纳此类型 VM 的数量。

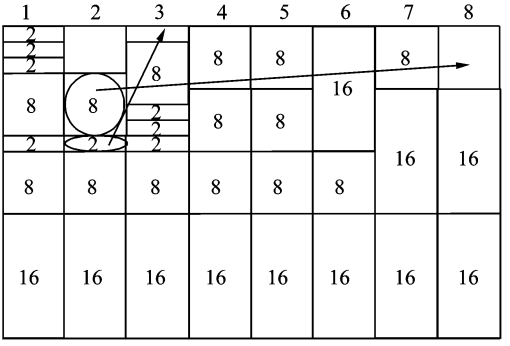
图 2 为执行 BVF 的例子,其中 $C=3$,$r_1=16$,$\phi_1=0.2$,$r_2=8$,$\phi_2=0.5$,$r_3=2$,$\phi_3=0.3$,$O_j=40$,$\forall j\in[1,8]$。图 2(a)所示为当前时间点 t 的网络资源状态,即 A^{t-},且假设 $n'=10$。下面执行 BVF 子算法依次统计 n_1^t,n_2^t,n_3^t:图 2(b)所示由于存在资源碎片,$n_1^t=0$。接下来遍历所有服务器尝试进行类型-2 的 VM 初始放置,有 $n_2^t=n'\cdot\phi_2=5$。最后尝试进行类型-3 的 VM 初始放置,有 $n_3^t=n'\cdot\phi_3=3$。因此,在此例中,执行完 BVF 子算法后,得到 $n_1^t=0<n'\cdot\phi_1$,$n_2^t=n'\cdot\phi_2$ 以及 $n_3^t=n'\cdot\phi_3$。



(a) 现有VM
(a) Existing VMs



(b) 执行BVF后：n_1^t=0, n_2^t=5, n_3^t=3
(b) After executing BVF：n_1^t=0, n_2^t=5, n_3^t=3



(c) 执行一次MinCM后
(c) After executing one time of MinCM

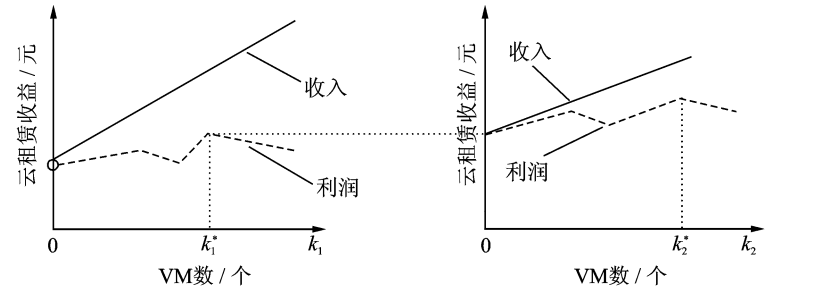
图 2　BVF 子算法举例示意图

Fig. 2　Diagram of illustration of BVF sub-algorithm

最大利润碎片整理(Maximum profit defragmentation,MaxPD)子算法:若 $\sum_{c=1}^{C}n_c^t=n'$,则算法终止;否则从满足条件 $n_c^t<n'_c=\phi_c\times n'$ 的最小类型索引号 c 开始:在每次迭代过程中,基于上次迭代所取得的最佳分配矩阵,通过迁移 VM 以及放置 k_c^* 个类型-c 的 VM,尝试得到最大利润以及本次迭代的最佳分配矩阵。

如图 3 所示,执行 BVF 子算法后,随着后续放入服务器中某类型 VM 数的增加,收入将提升。但由于迁移成本的引入,整体利润不一定呈线性增长趋势。因此采用 MaxPD 子算法为某类型 VM 寻求能取得最大利润的 k_c^* 值。

执行 BVF 子算法后,若后续容纳了 $k(0\leqslant k\leqslant n'_c-n_c^t)$ 个类型 c 的 VM,采用矩阵 $\boldsymbol{A}^{t,\alpha}(c,k)$ 和 $\boldsymbol{A}^{t',\beta}(c,k)$ 分别记录现有 VM 的迁移状态,以及新到达的、类型-c 的 VM 分配情况。根据矩阵 $\boldsymbol{A}^{t,\alpha}(c,k)$ 和



(a) 放置类型-1 VM情况下的收入与利润变化情况
(a) Variations of revenue and profit when we place type -1 VMs

(b) 放置类型-2 VM情况下的收入与利润变化情况
(b) Variations of revenue and profit when we place type -2 VMs

图 3 PRD 算法举例示意图

Fig. 3 Diagram of illustration of PRD algorithm

$\boldsymbol{A}^{t',\beta}(c,k)$的记录，可分别计算出收入$\overline{\pi}_{c,k}$和成本$\kappa_{c,k}$

$$\overline{\pi}_{c,k}=\overline{\pi}_{c-1,k_{c-1}^*}+k\times e_c \tag{20}$$

$$\kappa_{c,k}=\sum_{i=1}^{m_r}\sum_{j=1}^{S}\{l_r\cdot r_c\cdot a_{ij}\cdot[1-a(c,k)_{ij}^{a}]\} \tag{21}$$

式中，$k_0^*=0$；a_{ij}和$a(c,k)_{ij}^{a}$分别为$\boldsymbol{A}^{t-}$和$\boldsymbol{A}^{t,a}(c,k)$的矩阵元素。因此，$k_c^*$为对应最大利润$v_{c,k}=\overline{\pi}_{c,k}-\kappa_{c,k}$的$k$值。在每次执行 MaxPD 子算法迭代过程中，需连续调用最小成本迁移(Minimum-cost migration，MinCM)子算法，以实现对类型-c 的 VM 放置，直至网络资源不再充足或完成$(n_c'-n_c^t)$个类型-c 的 VM 放置。

MinCM 子算法：为避免迁移的乒乓效应，采用单向迁移技术保证迁移后的 VM 不再迁回原服务器，则有以下两个引理成立。

引理 1 存在参数c^*，使一个以上的、类型-c^*的 VM 无法采用 BVF 直接进行放置，且有$\forall c>c^*$，$n_c^t=n_c'$。

引理 2 执行 BVF 子算法后，若存在未能成功放置的 VM，则有$\forall j(1\leqslant j\leqslant S)$，$F_j<r_{c^*}$。

导出c^*后，初始化两种服务器集合：(1)存放迁出 VM 的服务器集合G_s；(2)具有可用剩余资源的服务器集合G_t。引入迁移控制，即仅有类型-$c'(c^*<c'\leqslant C)$的 VM 可被迁移，从而有

$$G_t=\{j\mid F_j\in[r_C,r_{c^*})\} \tag{22}$$

$$G_s=\{j\mid \exists\,\text{type-}c'\text{VM},c^*<c'\leqslant C\} \tag{23}$$

如图 2(b)所示，有$c^*=1$，且每个服务器的可用剩余资源均小于 16。因此，$G_s=\{1,2,3,4,5,6\}$，$G_t=\{2,3,6,7,8\}$。

为进一步容纳一个类型-c 的 VM，在G_s中确定一个服务器，通过将其内 VM 以最小的开销迁移到G_t中其他服务器，以腾出更多空间。具体地，首先在G_s中确定具有最大空闲空间的服务器j^*，若$j^*\in G_t$，更新$G_t'=G_t-\{j^*\}$。其次判断服务器j^*能否通过 VM 迁移后腾出空间容纳一个类型-c 的 VM：(1)计算服务器j^*中类型-c'的 VM 数$f_{c'}(j^*)$，$c^*<c'\leqslant C$；(2)采用 BVF 方法计算G_t'内所有服务器可用剩余资源最多可容纳的类型-c'的 VM 数$f_{c'}(G_t')$，$c^*<c'\leqslant C$；(3)确定从服务器j^*中最多能够迁出的、类型-c'的 VM 数$f_{c'}=\min[f_{c'}(j^*),f_{c'}(G_t')]$，$c^*<c'\leqslant C$；(4)确定能够容纳一个类型-$c$ 的 VM 的条件为

$$\sum_{c'=c^*+1}^{C}(f_{c'}\cdot r_{c'})+F_{j^*}\geqslant r_c \tag{24}$$

若满足上述条件,采用首次命中(First-fit,FF)方法将服务器 j^* 中的 VM 迁出;否则使 $G_s=G_s-\{j^*\}$,进行 while 循环直到 G_s 为空。例如,图 2(b)中可确定 $j^*=6$。根据上述定义可得 $f_2(6)=2$,$f_3(6)=0$。由于服务器 6 属于 G_t,则更新 $G_t'=\{2,3,7,8\}$,因此有 $f_2(G_t')=2$,$f_3(G_t')=4$。显然,$f_2=2$ 且 $f_3=0$。由于 $F_6+f_2\cdot r_2=8+8\times2=24>r_1=16$,从而将一个类型-2 的 VM 从服务器 6 迁移到服务器 7 后,将一个类型-1 的 VM 放入服务器 6 中,如图 2(b)所示。采用类似方法,可确定 $j^*=2$,并通过将一个类型-2 的 VM 从服务器 2 迁移到服务器 8,同时将一个类型-3 的 VM 从服务器 2 迁移到服务器 3,进一步将一个类型-1 的 VM 放入服务器 2 中,如图 2(c)所示。

PRD 算法时间复杂度主要取决于运行 FF 方法的次数,而每运行一次 FF 方法的时间复杂度为 $C^2\cdot S$。在 MinCM 子算法中最多调用 S 次 FF 方法,而在主算法中最多调用 n' 次 MinCM 子算法,因此,PRD 算法时间复杂度约为 $O(n'\cdot C^2\cdot S^2)$。

3 仿真实现与性能分析

采用异构云数据中心网络,即服务器总容量可从{40,60,80}中随机选取。引入时间轴,该轴被等分为 24 个时隙。VM 请求在某一时隙的起始端到达,且服务持续时间占用时隙数满足均匀整数分布[1,4]。网络内存在 $C=3$ 种类型 VM:$r_1=16$,$r_2=8$,$r_3=2$ 且 $e_c=r_c$,$c=1,2,3$。给定 $\Phi=\{0.2,0.5,0.3\}$和以下仿真场景。

场景 1 $S=35$,$l_r=0.01$,预测未来到达 VM 请求数采用均值$\overline{n}$,依次取为{150,200,250,300,350},且 $n'=\overline{n}$(即预测精确度为 100%)。

场景 2 $S\in\{35,40,45,50,55\}$,$l_r=0.01$,均值$\overline{n}$从 150 增长到 300,即每 5 个时隙后$\overline{n}$增长 50,但 $n'=230$(即存在预测误差)。

总利润(Total profit,TP):每个时隙产生的利润为相应收入与迁移成本之差,总利润为 20 个时隙产生利润总和;平均迁移成本(Average migration cost,AMC)为 20 个时隙产生的总迁移成本与执行碎片整理操作的次数之比。

表 1,2 分别显示在场景 1 和 2 下可为固定周期的服务器整合方法找到最佳周期 T,使相应总利润最大(最大利润为表中粗体数据)。在场景 2 下,各方法的总利润随着服务器数量的增加而升高。同时,所提 PRD 算法的总利润高于未采用碎片整理的普通 VM 放置方法和固定周期的服务器整合方法,即使在预测精度较低情况下,PRD 总利润同样接近于总利润最优值,偏差率仅为 6%。

表 1 场景 1 下的总利润比较结果

Table 1 Comparative results of TP under scenario 1

$\overline{n}$	150	200	250	300	350
普通 VM 放置方法	16 434	16 816	16 806	17 036	16 858
$T=1$	16 473.26	16 818.5	16 797.88	17 042.86	**16 870.78**
$T=2$	16 495.44	16 821.16	16 761.12	17 033.34	16 865.18
$T=3$	16 410.06	16 759.5	16 782.82	16 975.12	16 796.08
$T=4$	**16 504.72**	16 822.58	16 763.56	17 019.18	16 822.92
$T=5$	16 358.52	**16 834.72**	**16 823.96**	17 006.18	16 796.24
$T=6$	16 442	16 806.36	16 788.26	17 027.28	16 831.82
$T=10$	16 472	16 828.24	16 812.18	17 032.92	16 847.4
$T=20$	16 445.22	16 826.18	16 816.26	**17 047.44**	16 867.06
PRD	16 608.16	16 989.6	17 003.7	17 241.74	17 063.72
最佳值	18 064.8	17 518.8	17 277	17 495.4	17 300.4

表 2 场景 2 下的总利润比较结果

Table 2 Comparative results of TP under scenario 2

S	35	40	45	50	55
普通 VM 放置方法	16 592	18 480	20 312	22 648	24 536
$T=1$	**16 657.06**	18 447.62	20 371.24	**22 705.54**	**24 685.56**
$T=2$	16 635.16	18 495.72	20 373.46	22 601.78	24 622.6
$T=3$	16 638.62	18 452.74	20 297.6	22 595.84	24 517.48
$T=4$	16 636.7	18 482.76	**20 400.44**	22 651.26	24 624.98
$T=5$	16 586.32	18 511.26	20 342.14	22 691.44	24 645.44
$T=6$	16 598.16	18 468.18	20 300.54	22 690.84	24 584.68
$T=10$	16 629.2	**18 516.22**	20 329.46	22 664.28	24575.88
$T=20$	16 603.42	18 496.86	20 329.18	22 664.12	24 552.22
PRD	16 789.76	18 677.68	20 533.44	22 821.92	24 710.1
最佳值	17 846.4	20 085	22 815	26 075.4	29 117.4

本文假设网络内服务器可用资源可整合成连续的、无碎片的资源池，即可得到总利润最优值。

图 4(a,b)分别显示在场景 1,2 下 PRD 算法有效地降低了平均迁移成本，且 PRD 平均迁移成本不会随着 VM 请求数和服务器数量的增加而发生明显变化。采用固定周期的服务器整合方法所产生的迁移成本会随着服务器数量的增加而线性增长，这意味着 PRD 算法更加灵活，且适用于大规模的云数据中心网络。

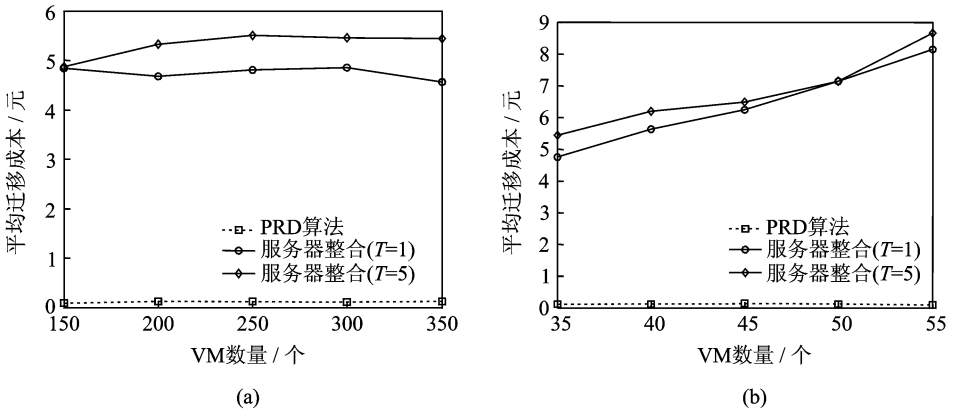


图 4 平均迁移成本比较结果示意图

Fig. 4 Diagram of comparative results of migration cost

4 结束语

虚拟机的动态创建和下线会使云数据中心网络内产生大量的资源碎片，本文给出了针对云数据中心网络资源碎片整理的系统阐述和分析。首先提出了在某一时段最小化无效 VM 迁移成本的优化问题，并对其进行数学定义。随后设计了启发式算法获取上述问题的近似最优解，并通过仿真结果验证了

所提算法的优越性。本文所设计的算法只考虑了单一维度的计算资源。因此在未来的研究工作中,将重点考虑设计面向多维度计算资源的碎片整理算法。此外,多维度资源(如计算、内存、存储、带宽以及I/O等)之间的相关性也将是未来设计碎片整理算法所要考虑的重要方面。

参考文献:

[1] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing[J]. Commun ACM,2010,53(4):50-58.

[2] Sridharan M, Calyam P, Venkataraman A, et al. Defragmentation of resources in virtual desktop clouds for cost-aware utility optimal allocation[C]// Proceedings of IEEE International Conference on Utility and Cloud Computing. Los Alamitos: IEEE Computer Society Press, 2011:253-260.

[3] He S, Guo L, Guo Y. Real time elastic cloud management for limited resources[C]// Proceedings of IEEE International Conference on Cloud Computing. Los Alamitos: IEEE Computer Society Press, 2011:622-629.

[4] Mishra M, Sahoo A. On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach[C]// Proceedings of IEEE International Conference on Cloud Computing. Los Alamitos: IEEE Computer Society Press, 2011:275-282.

[5] He S, Guo L, Ghanem M, et al. Improving resource utilisation in the cloud environment using multivariate probabilistic models[C]// Proceedings of IEEE International Conference on Cloud Computing. Los Alamitos: IEEE Computer Society Press, 2012:574-581.

[6] Calcavecchia N, Biran O, Hadad E, et al. VM placement strategies for cloud scenarios[C]// Proceedings of IEEE International Conference on Cloud Computing. Los Alamitos: IEEE Computer Society Press, 2012:852-859.

[7] Khanna G, Beaty K, Kar G, et al. Application performance management in virtualized server environments [C] // Proceedings of Network Operations and Management Symposium. Piscataway: IEEE, 2006:373-381.

[8] Bobroff N, Kochut A, Beaty K. Dynamic placement of virtual machines for managing SLA violations[C]// Proceedings of International Symposium on Integrated Network Management. Piscataway: IEEE, 2007:119-128.

[9] Speitkamp B, Bichler M. A mathematical programming approach for server consolidation problems in virtualized data centers [J]. IEEE Transactions on Services Computing, 2010,3(4):266-278.

[10] Wang M, Meng X, Zhang L. Consolidating virtual machines with dynamic bandwidth demand in data centers[C]// Proceedings of IEEE INFOCOM. Los Alamitos: IEEE Computer Society Press, 2011:71-75.

[11] Wood T, Shenoy P, Venkataramani A, et al. Black-box and gray-box strategies for virtual machine migration[C]// Proceedings of the 4th USENIX conference on Networked systems design implementation. Piscataway: IEEE, 2007:17-17.

[12] Xu L, Chen W, Wang Z, et al. Smart-DRS: A strategy of dynamic resource scheduling in cloud data center[C]// Proceedings of International Conference on Cluster Computing Workshops. Piscataway: IEEE, 2012:120-127.

[13] Beloglazov A, Buyya R. Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints[J]. IEEE Transactions on Parallel and Distributed Systems, 2013,24(7):1366-1379.

[14] Deng W, Liu F, Jin H, et al. Lifetime or energy: Consolidating servers with reliability control in virtualized cloud datacenters[C]// Proceedings of International Conference on Cloud Computing Technology and Science. Los Alamitos: IEEE Computer Society Press, 2012:18-25.

[15] Feller E, Morin C, Esnault A. A case for fully decentralized dynamic vm consolidation in clouds[C]// Proceedings of International Conference on Cloud Computing Technology and Science. Los Alamitos: IEEE Computer Society Press, 2012:26-33.

作者简介:

郭磊(1980-),男,教授,研究方向:光网络、无线通信,E-mail: haveball @ gmail.com。

侯维刚(1984-),通信作者,男,副教授,研究方向:光数据中心网络、云计算、虚拟化,E-mail: houweigang @ ise. neu. edu. cn。