



云计算中一种高效的虚拟机在线动态分配算法

张丽敏

(西安外事学院现代教育技术中心 西安 710077)

摘要:设计了一种改进的 VM 在线动态供应和分配算法(ODAA-VM)。只要用户发现请求或已经被分配的部分 VM 实例再次可用,便迅速调用 ODAA-VM,确定在所要求的时期内为哪些用户分配 VM 实例,并保证这些用户在整个被请求期间可以持续使用这些 VM。此外,该算法对用户具有激励作用,仿真实验验证了本文算法的有效性。

关键词:云计算;在线算法;组合拍卖;分配;激励

doi: 10.11959/j.issn.1000-0801.2015048

Efficient Online Dynamic Allocation Algorithm of Virtual Machine in Cloud Computing

Zhang Limin

(The Modern Educational Technology Center, Xi'an International University, Xi'an 710077, China)

Abstract: An improved online dynamic allocation algorithm of VM (ODAA-VM) instances was proposed. The algorithm was invoked as soon as a user places a request or some VM instances already allocated become available again, then the algorithm selects users who would be allocated VM instances for the period they requested and ensures that those users would continue using those VM for the entire period requested. In addition, the proposed algorithm has the incentive effects for the user. Simulation experiments validate the effectiveness of the proposed algorithm.

Key words: cloud computing, online algorithm, combinatorial auction, allocation, incentive

1 引言

虚拟技术为云供应商提供更加便捷的方法来分配计算资源。云供应商定义不同的虚拟机(VM)组合,并以 VM 实例为单位出售资源。当前,云供应商使用定价策略分配和出售 VM 实例^[1],使用拍卖算法来出售定价之后仍然闲

置的资源。云供应商希望分配算法能够支持动态供应,以便他们根据市场需求就不同类型的 VM 实例数量做出决策。云供应商也希望实现营运收入最大化。

多篇文献已经对云计算的经济问题进行了研究。参考文献[2]针对目前国内并没有相应的云计算计价标准,价格高低直接影响网络利用率和收益的问题,在计价时考虑价

收稿日期:2014-09-29;修回日期:2015-01-28

基金项目:陕西省教育厅 2014 年科学研究项目(No.14JK2117)

Foundation Item: Scientific Research Fund of Shaanxi Provincial Education Department in 2014(No.14JK2117)

论文引用格式:张丽敏. 云计算中一种高效的虚拟机在线动态分配算法. 电信科学, 2015048

Zhang L. M. Efficient online dynamic allocation algorithm of virtual machine in cloud computing. Telecommunications Science, 2015048



格因素对网络负载的影响,提出了一种改进的蚁群算法,解决了计算资源闲置导致的网络利用率低下和收益减少问题。参考文献[3]提出了一种面向资源分配的第 $k+1$ 个价格拍卖算法。通过理论分析与仿真得出,当资源和用户较多时,该算法既可以提高分配的效率,又可以增加云服务商的收入。参考文献[4]设计了一种基于马尔可夫决策过程的资源在线算法,给出了同一模型面对大规模问题时的近似求解方法。参考文献[5]设计了一种并行系统下可拓展并行作业的资源在线调度算法。该算法在其模型中支持作业抢占,如果作业的价值更高,那么即使提交时间晚于当前分配的作业,也可以获得优先权支持。参考文献[6]提出了一种云资源调度模型(cloud-resource scheduling model, CSM),该模型以实现效用最大为调度目标,可以充分提高用户的满意程度。文中通过求解CSM优化问题得到最优的虚拟机和物理机映射关系,实现了对云资源的最优分配。

另外,参考文献[7]针对云商务环境下信息服务资源分配过程中的定价问题,分别建立了服务者、服务用户和服务环境的模型,并提出相应的求解方法。仿真实验结果表明,当交易者的规模增加时,该方法可以在保证第三方收入的情况下,使参与交易的双方获取更高的服务效用。参考文献[8]提出了一种基于组合拍卖的VM分配算法CA-Provision,该算法通过假设资源以数量固定但类型不同的VM实例形式提前供应,从而解决VM分配问题。另外,CA-Provision算法拓展了参考文献[7]中的模型,并支持VM实例的动态供应和分配。然而,该算法在运行时需要收集一段时间内所有用户请求的所有信息,即这些算法是离线算法,需要定期运行来为用户分配资源。

为此,本文设计了一种云条件下VM实例的在线供应和分配算法。首先,设计了一种申请协议,当用户申请一批VM时,表明用户只对整批VM感兴趣,而不是其中的一部分。在接收到这些申请后,本文算法在线计算分配策略和花费。同时,分配不得被抢占,如果用户在时间 t 接收到分配策略,则在用户所请求的该段时间内始终占有资源。最后,通过理论分析和仿真实验验证了本文算法的有效性。

2 VM实例分配问题

假设云服务提供商提供的计算资源共有 m 种不同的VM实例类型,本文将这些实例类型表示为 $VM_1, \dots, VM_m$ 。对每种类型的实例 VM_i 关联一个“权重” $\omega_i \in \mathcal{R}_+^*$, ω_i 表示 VM_i 相对云供应商提供的最强大VM实例(即类型为 VM_1

的实例)的计算能力。不失一般性,本文认为 $\omega_1=1$ 且 $\omega_1 \leq \dots \leq \omega_m$ 。将可以用于分配的可用计算资源的总计算能力表示为 M ,定义为类型 VM_i 的 M 个实例的等价计算能力。假设云供应商提供3种类型的VM实例:小型(VM_1)、中型(VM_2)、大型(VM_3)。这些VM的配置如下: VM_1 为1个2 GHz处理器、8 GB内存、2 TB硬盘; VM_2 为2个2 GHz处理器、8 GB内存、2 TB硬盘; VM_3 为2个2 GHz处理器、16 GB内存、4 TB硬盘。对于该配置来说,VM的权重为: $\omega_1=1, \omega_2=2, \omega_3=4$ 。假设云提供商有足够的资源来提供100 000个 VM_1 实例,则 $M=100\,000$ 。云提供商的目的是将其可用资源动态供应为VM实例,在将其高效提供给用户的同时实现收益最大化。

设有 $j(j=1, 2, \dots, n)$ 个用户向云提供商发出申请,希望获得一定时间内一批VM实例的使用权,以便运行云中的应用(作业)。用户 j 的类型为 $\theta_j=(r_1^j, \dots, r_m^j, a_j, l_j, d_j, v_j) \in \Theta_j$,其中 Θ_j 是其类型空间, v_j 为系统受益, $(r_1^j, \dots, r_m^j)$ 是用户 j 请求的一批VM实例, $r_i^j \in N$ 是用户 j 请求的 VM_i 实例的数量。将一批实例表示为 S_j ,其总体权重为 $s_j = \sum_{i=1}^m \omega_i r_i^j$ 。用户 j 的类型 θ_j 的元素包括:到达时间(即用户 j 发出请求的时间) $a_j \in N$;被请求的批量实例必须被分配的时间 $l_j \in N$;用户 j 的作业必须完成的截止时间 $d_j \in N$ 。 δ_j 表示为了完成作业需为用户 j 分配的时间,即 $\delta_j = d_j - l_j$ 。 θ_j 的最后一个元素为 $v_j \geq 0$,表示用户 j 在 δ_j 内分配到了所请求的批量VM。假设用户为“专一型”用户,即用户 j 要求:要么请求的批量实例 S_j 分配成功,自己支出费用;要么没有分配到任何资源,自己完全不用支付费用。

云提供商通过动态供应可用资源,将VM批量实例分配给用户,同时收取费用。提供商必须在时间 $[a_j, d_j]$ 内就批量实例 S_j 的分配问题做出决策。如果用户在该时间段内没有被分配资源,则需要拒绝用户申请,同时用户收回申请。最后,云提供商计算分配集合 $A \subseteq \Theta \times N$ 及费用集合 $P \subseteq \Theta \times \mathcal{R}_+$ 。元组 $(\theta_j, t_j) \in A$ 表示在时间 t_j 为用户 j 分配了类型为 θ_j 的批量实例 S_j ,且 $a_j \leq t_j \leq d_j$ 。元组 $(\theta_j, p_j) \in P$ 表示类型为 θ_j 的用户 j 必须向云提供商支付费用 p_j 。对上述用户 j ,如果在时间 $t=3$ s内被分配到资源,并要价30美元,则有 $(\theta_j, 3) \in A$ 且 $(\theta_j, 30) \in P$;如果用户没有分配到所申请的资源,则不需要支付任何费用。将VM在线供应和分配问题定义为:

$$\max \sum_{j: (\theta_j, t_j) \in A} v_j \quad (1)$$

满足的约束条件为:

$$\sum_{j: \theta_j \in \tilde{N}^{(t)}} S_j \leq M \quad (2)$$

式(1)表示问题的目标,即云供应商使成功申请资源的用户价值之和最大化。式(2)表示在已知时间 t 分配的资源不得超过最大可用资源数量 M 。其中, $\tilde{N}^{(t)} = \{j | \exists t_j \leq t: ((\theta_j, t_j) \in A) \cap (t_j + l_j > t)\}$, 该集合表示在时间 t 及 t 之前分配到资源,并且在时间 t 所分配到的时间间隔还没有结束的用户集合。

上述问题的一种直接求解方法就是选择价值最大的用户,然后向这些用户收取各自的费用。然而,用户是理性用户,他们可能为了自身利益而误报自己的类型。例如,用户可能少报价值以便减少支付额度,或者多报价值以增加被分配概率。如果云提供商偏好截止日期较早的作业而忽略价值,则用户可能会早报他们的截止日期。因为这一信息是隐私信息,所以提供商需要使用一种算法来根据用户报告的价值计算分配策略及相关费用,以便提供商设定的全系统目标能够实现。

因此,VM 在线供应和分配问题的在线算法如下:设计一种可在问题空间 $\Theta = \Theta_1 \times \cdots \times \Theta_n$ 计算出分配集合 A 和费用集合 P 的在线算法,在满足式(2)约束条件的同时实现式(1)目标函数的最大化。用户提供给算法的信息为 $\hat{\theta} = (\hat{\theta}_1, \cdots, \hat{\theta}_n) \in \Theta$, 其中 $\hat{\theta}_j \in \Theta_j$ 。在线算法的目标是:即使 $\hat{\theta}_j \neq \theta_j$,也要计算出一种高效的分配策略,并计算出相关费用,以便促使用户汇报自己的真实类型。

3 在线算法的设计

算法 $E = \{A, P\}$ 包括用于计算分配策略的一组函数 A 及用于计算每个用户费用的一组函数 P 。其中 $A: \Theta \rightarrow A$, 且 $P = (p_1(\cdot), \cdots, p_n(\cdot))$ 。对于 $j=1, \cdots, n$, 有 $p_j: \Theta \rightarrow \mathcal{R}$ 。对于 OVMPA 问题, 分配函数 A 根据用户的申请来计算分配函数 $A \subseteq \Theta \times N$ 。分配集合 A 表示元组 (θ_j, t_j) 集合, 其中 j 表示在时间 t_j 接收到所申请的批量实例的用户。函数 $p_j(\cdot)$ 可根据所有用户的申请来确定用户 j 的费用。

每个用户 j 可将价值函数 V_j 定义如下:

$$V_j(A(\hat{\theta}), \theta_j) = \begin{cases} v_j, & (\theta_j, t_j) \in A \wedge t_j \leq d_j' \\ 0, & \text{其他} \end{cases} \quad (3)$$

如果用户申请成功,则用户接收到的价值为 v_j ; 否则,接收到的价值为 0。本文用效用函数来定量描述用户 j 的

效益,该效用函数定义为从算法接收到的价值与用户需要支付的费用之差,即:

$$U_j(A(\hat{\theta}), \theta_j) = V_j(A(\hat{\theta}), \theta_j) - p_j(\hat{\theta}) \quad (4)$$

如果在线算法拒绝为其分配资源,则用户接收到的价值为 0。此时,如果算法不向其收取任何费用,用户的效用也将为 0。

设计的在线算法有两大属性:激励作用及个体理性。用 $\hat{\theta}_{-j} = (\hat{\theta}_1, \cdots, \hat{\theta}_{j-1}, \hat{\theta}_{j+1}, \cdots, \hat{\theta}_n) \in \Theta_{-j}$ 表示除了用户 j 的所有用户申请。用户 j 的真正类型为 θ_j , 而该用户所汇报的类型(或申请) $\hat{\theta}_j$ 可能与 θ_j 不一致。

定义 1 (激励作用) 如果对任意用户 j 有 $\hat{\theta}_j \in \Theta_j$ 、 $\hat{\theta}_{-j} \in \Theta_{-j}$ 、并满足式(5),则算法 E 具有激励作用^[9]。

$$U_j(A(\theta_j, \hat{\theta}_{-j}), \theta_j) \geq U_j(A(\hat{\theta}_j, \hat{\theta}_{-j}), \theta_j) \quad (5)$$

用户如果不管其他用户的申请如何,而把自己的真实类型汇报给算法,则可以实现自身效用最大化。这是非常重要的算法,因为如果满足这一属性,参与算法的用户就不会向算法汇报虚假类型,即汇报真实属性是用户的最优策略。

定义 2 (个体理性) 如果用户汇报了真实类型后不会遭受损失,则称算法具有个体理性。即对于任意用户 j 、真实类型 $\hat{\theta}_j \in \Theta_j$ 及 $\hat{\theta}_{-j} \in \Theta_{-j}$, 有:

$$U_j(A(\theta_j, \hat{\theta}_{-j}), \theta_j) \geq 0 \quad (6)$$

不管其他用户的申请如何,用户只要参与在线算法并汇报了自己的真实类型,自己的效用一定不会为负。这也是在线算法的重要属性,因为可以鼓励用户积极参与在线算法。为了使算法具有激励作用,分配函数 A 必须单调且费用函数 P 必须实现临界值支付^[10]。

对用户申请集合引入偏好关系“ $\geq$ ”。例如, $\hat{\theta}_j' \geq \hat{\theta}_j$ 表示相比于申请 $\hat{\theta}_j$, 算法更偏好于申请 $\hat{\theta}_j'$ 。在本文中,如果 $\hat{s}_j' \leq \hat{s}_j$, $\hat{a}_j' \leq \hat{a}_j$, $\hat{l}_j' \leq \hat{l}_j$, $\hat{d}_j' \geq \hat{d}_j$, 且 $\hat{v}_j' \geq \hat{v}_j$ 则 $\hat{\theta}_j' \geq \hat{\theta}_j$ 。

定义 3 (单调分配函数) 如果对任意 j 、 $\hat{\theta}_j$ 、 $\hat{\theta}_j' \in \Theta_j$ 、 $\hat{\theta}_{-j} \in \Theta_{-j}$ 且式(7)成立,则分配函数 A 单调。

$$\begin{aligned} & (\hat{\theta}_j' \geq \hat{\theta}_j) \wedge (\exists t_j: (\hat{\theta}_j, t_j) \in A(\hat{\theta}_j, \theta_{-j})) \\ & \Rightarrow \exists t_j': (\hat{\theta}_j', t_j') \in A(\hat{\theta}_j', \theta_{-j}) \end{aligned} \quad (7)$$

如果用户 j 通过汇报类型 $\hat{\theta}_j$ 而申请成功,则该用户汇报 $\hat{\theta}_j'$ 也会申请成功,其中 $\hat{\theta}_j'$ 比 $\hat{\theta}_j$ 更受偏好。

定义 4 (临界值支付) 用户 j 的临界值 V_j^c 定义为



$$v_j^c = \arg \lim_{v_j' \geq 0} (\exists t_j : (\hat{\theta}_j = (r_1^j, \dots, r_m^j, a_j, l_j, d_j, v_j'), t_j) \in A(\hat{\theta}_j, \theta_{-j})).$$

临界值是指用户为了申请成功而必须向算法汇报的最小值。费用函数应该向用户收取临界值费用,以便使算法具有激励功能。本文设计了一种解决 OVMPA 问题的在线算法。

4 在线 VM 分配算法

本节给出了解决 OVMPA 问题的在线 VM 分配和供应算法。

4.1 ODAA-VM

ODAA-VM 采取事件处理程序架构,当新的申请到达或者用户分配到的批量资源到期,并且将 VM 实例返回给供应商时,调用事件处理程序。假设通过某种标准化协议,用户和资源的信息对算法是已知的。ODAA-VM 利用这一信息来确定用户集合及当前时间可用资源,并可调用分配函数和费用函数。ODAA-VM 流程如下。

输入: $Event, A, P$ 。

输出: A, P 。

(1) $t \leftarrow$ 当前时间

(2) $N^{(t)} \leftarrow \{\hat{\theta}_j \in N \mid \hat{a}_j \leq t \wedge \neg \exists t_j < t : (\hat{\theta}_j, t_j) \in A\}$

(3) $\tilde{N}^{(t)} \leftarrow \{\hat{\theta}_j \mid \exists t_j < t : (\hat{\theta}_j, t_j) \in A \wedge t_j + \hat{l}_j > t\}$

(4) $M^{(t)} \leftarrow M - \sum_{j: \hat{\theta}_j \in \tilde{N}^{(t)}} \hat{s}_j$

(5) if $M^{(t)} = 0 \vee N^{(t)} = \emptyset$ then return

(6) $R \leftarrow M^{(t)}$

(7) $A^{(t)} \leftarrow \text{MOVMPA-Allocate}(t, N^{(t)}, R)$

(8) $A \leftarrow A \cup A^{(t)}$

(9) $P^{(t)} \leftarrow \{(\hat{\theta}_j, \hat{v}_j) \mid (\hat{\theta}_j, t_j) \in A^{(t)}\}$

(10) $P \leftarrow P \cup P^{(t)}$

(11) $P \leftarrow \text{MOVMPA-Payment}(t, P, N^{(t)}, \tilde{N}^{(t)}, M^{(t)})$

(12) $N^P \leftarrow \{\hat{\theta}_j \in N \mid t' < \delta_j \leq t\}$

(13) for 每个 $\hat{\theta}_j \in N^P$ do

(14) if $\exists p_j : (\hat{\theta}_j, p_j) \in P$ then

(15) 用户 j 支付 p_j

(16) else

(17) 用户 j 支付 0

(18) end if

(19) end for

(20) $t' \leftarrow t$

(21) 返回 A, P

ODAA-VM 以事件、当前分配集合及费用集合为输入。事件类型包括资源的释放或到达的用户申请。假设系统存储这两个集合,并且当 ODAA-VM 被调用时将其传输给 ODAA-VM。ODAA-VM 更新集合,然后返回给系统。在第 1~4 行, ODAA-VM 设置当前时间为 t ,且 3 个变量的初始化如下: $N^{(t)}$ 为迄今还没有申请成功的用户申请集合; $\tilde{N}^{(t)}$ 为已经成功申请但是分配的资源还没有到期的用户申请集合; $M^{(t)}$ 为在时间 t 可用于分配的资源总权重。只有当资源和用户可用时,算法才会运行。算法会调用以未获得成功的用户申请和时间可用资源为参数的函数 ODAA-VM-Allocate。ODAA-VM-Allocate 函数(见第 4.2 节)返回在时间 t 申请成功的用户集合 $A^{(t)}$ 。

然后, ODAA-VM 将整个分配集合 A 更新为 $A^{(t)}$ 。 $A^{(t)}$ 中的申请也被插入费用集合,且 $p_j = \hat{v}_j$ 为初始费用。然而,通过调用 ODAA-VM-Payment 函数(见第 4.3 节)可以更新该费用(第 11 行)。实际上,用户 j 的费用会被更新多次,直到 $t = \delta_j$ (即用户接收到所要求的批量实例)为止。因此, ODAA-VM-Payment 函数计算这些用户的费用并更新费用集合 P 。ODAA-VM 算法的下一个步骤是确定当前时间已经超过 δ_j 的用户 j 的申请集合 N^P (第 12 行)。然而,这一集合只包括 t' 之后已经度过 δ_j 时间的用户,其中 t' 表示上次调用 ODAA-VM 的时间(第 12 行)。如果用户 j 已经分配到了批量资源,则该用户的费用不会再改变,当前时刻的费用即最终费用。如果用户 j 直到时间 t 还没有申请成功且 $t > \delta_j$,则用户不会申请到完成他的作业所需要的资源。此时,用户 j 将支付的费用为 $p_j = 0$ (第 14~19 行)。

4.2 ODAA-VM-Allocate 函数

函数 ODAA-VM-Allocate 的具体过程如下。

输入: $t, N^{(t)}, R$ 。

输出: $A^{(t)}$ 。

$A^{(t)} \leftarrow \emptyset$

按照 p_j 的非升次序排列所有的 $\hat{\theta}_j \in N^{(t)}$

for 排序过后的每个 $\hat{\theta}_j \in N^{(t)}$ do

if $s_j \leq R$ then

$A^{(t)} \leftarrow A^{(t)} \cup (\hat{\theta}_j, t)$

$R \leftarrow R - \hat{s}_j$

end if

end for

返回 $A^{(t)}$

为了描述函数 ODAA-VM-Allocate, 定义“申请密度”的新参数 $\rho_j = \frac{\hat{v}_j}{\hat{s}_j \times \hat{l}_j}$ 。可以把 VM 分配问题表述为在 VM 权重和时间二维空间内的矩阵分配问题。将用户 j 为了在时间 $\hat{l}_j$ 内使用权重为 $\hat{s}_j$ 的 VM 实例而发出的申请看成申请二维空间内面积为 $\hat{s}_j \times \hat{l}_j$ 的一个矩形, 用户 j 对该面积的估价 $\hat{v}_j$ 。因此, ρ_j 表示用户 j 对该矩形区域“单位面积”的估价。很明显, 云供应商对愿意为单位时间、单位资源支出更多费用的用户感兴趣。ODAA-VM-Allocate 函数使用 ρ_j 来确定不同申请的相对价值。

首先, ODAA-VM-Allocate 函数按照 ρ_j 的非升次序对所有申请排序。按照如下次序打破关联: 偏爱更早的 δ_j 、更小的 $\hat{l}_j$ 以及更小的 $\hat{s}_j$ 。然后, 当资源允许时, 算法将为这些用户分配各自请求的资源。最后, 算法返回在时间 t 被选择分配资源的用户集合 $A^{(t)}$ 。

ODAA-VM-Allocate 函数的目的是实现申请成功的用户价值之和最大化。当存在关联情况时, 通过向 δ_j 较小的用户赋予优先级 (即没有申请成功而将较早离开系统的用户), 也可以保证受到服务的用户数量最大化。基于同样原因, 对于 δ_j 关联情况, 资源使用时间较短的用户也将受到优先对待。

4.3 ODAA-VM-Payment 函数

ODAA-VM-Payment 函数如下。

输入: $t, P, N^{(t)}, \tilde{N}^{(t)}, M^{(t)}$ 。

输出: P 。

(1) 按照 ρ_j 非升序次序排列所有 $\hat{\theta}_j \in N^{(t)}$

(2) for 每个 $(\hat{\theta}_j, p_j) \in P: \delta_j \leq t$ do

(3) $N'^{(t)} \leftarrow N^{(t)} / \hat{\theta}_j$

(4) if $\hat{\theta}_j \in \tilde{N}^{(t)}$ then

(5) $R' \leftarrow M^{(t)} + \hat{s}_j$

(6) else

(7) $R' \leftarrow M^{(t)}$

(8) end if

(9) for 每个 $\hat{\theta}_{j'} \in N^{(t)}$ do

(10) $\bar{v}_j \leftarrow \rho_{j'} \times \hat{s}_j \times \hat{l}_j$

(11) $\bar{\theta}_j \leftarrow (r_1^j, \dots, r_m^j, t, \hat{l}_j, \hat{d}_j, \bar{v}_j)$

(12) $A' \leftarrow \text{MOVMPA-Allocate}(t, \tilde{N}^{(t)} \cup \{\bar{\theta}_j\}, R')$

(13) if $(\bar{\theta}_j, t) \in A'$ then

(14) $p_j \leftarrow \min(p_j, \bar{v}_j)$

(15) end if

(16) end for

(17) $\bar{\theta}_j \leftarrow (r_1^j, \dots, r_m^j, t, \hat{l}_j, \hat{d}_j, 0)$

(18) $A' \leftarrow \text{MOVMPA-Allocate}(t, \tilde{N}^{(t)} \cup \{\bar{\theta}_j\}, R')$

(19) if $(\bar{\theta}_j, t) \in A'$ then

(20) $p_j \leftarrow 0$

(21) end if

(22) end for

(23) 返回 P

ODAA-VM-Payment 函数的输入包括当前时间 t 、费用集合 P 、调用分配函数 $M^{(t)}$ 之前的可用资源、分配函数 $N^{(t)}$ 考虑的用户集合以及当前时间 t 占据资源的用户集合 $\tilde{N}^{(t)}$ 。在时间 t 申请成功的用户也属于集合 $N^{(t)}$ 。

ODAA-VM-Payment 函数的主要思路是将用户的到达时间看成 t , 然后计算 $\delta_j \leq t$ 的用户 j 的临界费用。通过每个事件重复调用这一函数, ODAA-VM 可以保证 $\hat{a}_j$ 和 δ_j 之间每次出现事件时, 均会计算用户 j 的临界费用。ODAA-VM-Payment 函数对 $\delta_j \geq t$ 的所有用户 j 计算时间 t 时的临界值为:

$$v_j^c = \arg \min_{v_j' \geq 0} (\exists t_j : (\hat{\theta}_j' = (\hat{r}_1^j, \dots, \hat{r}_m^j, t, \hat{l}_j, \hat{d}_j, v_j'), t_j) \in A(\hat{\theta}_j, \theta_{-j})) \quad (8)$$

以该临界值为基础, ODAA-VM 将临界值计算为:

$$v_j^c = \min_{t \in [\hat{a}_j, d_j]} v_j^c \quad (9)$$

即只要到达时间 $a_j' \in [\hat{a}_j, d_j]$, 用户 j 为了申请成功, 其所汇报的数值必须大于或等于 v_j^c 。

ODAA-VM-Payment 函数考虑费用集合中 $\delta_j \leq t$ 的用户。对于每个用户 j , 将其类型的到达时间元素设为 t , 价值设为每个用户的价值, 以便确定用户 j 为了申请成功而需汇报的最小值 (第 2~16 行)。如果未找到最小值, 则将支付费用设为 0 (第 17~21 行), 也可以将该数值设为预先确定的底价。最后, 将更新过的集合 P 返回给在线算法。在线算法通过调用 ODAA-VM-Payment 函数以持续更新 P , 并在时间 t 向的 $\delta_j < t$ 用户 j 收取更新过的费用。没有申请成功的用户不需支付费用。



5 算法性能

定理 1 ODAA-VM 具有激励作用。

为了证明这一定理,需要证明分配函数是单调函数且费用函数支持临界值支付。首先,证明分配函数是单调的。很明显,如果用户 j 申请时有 $\hat{v}_j' \geq \hat{v}_j$, 那么若 $\hat{\theta}_j$ 被分配,则该用户也应该被分配。类似地,如果有用户汇报 $\hat{d}_j$ 后申请成功,则该用户汇报 $\hat{d}_j' \geq \hat{d}_j$ 时,也应该申请成功。然后,根据 ODAA-VM-Allocate 函数为了对汇报类型进行排序所设立的优先级(第 4.2 节),可知 ODAA-VM-Allocate 函数是单调的。如果其他用户汇报的类型保持不变,那么当计算用户为了申请成功而必须汇报最小价值时,该最小值对 $\hat{\theta}_j'$ 和 $\hat{\theta}_j$ 是相同的。因此,用户 j 汇报两种类型时的支付费用是相同的,即 ODAA-VM-Payment 函数支持临界值支付。

定理 2 ODAA-VM 的时间复杂度为 $O(n \log n)$ 。

ODAA-VM 主要包括 ODAA-VM-Allocate 函数和 ODAA-VM-Payment 函数。其中,ODAA-VM-Allocate 函数成本最大的操作是第 2 行的用户排序,这一操作的复杂度为 $O(n \log n)$ 。ODAA-VM-Payment 函数看上去非常复杂,因为它重复调用 ODAA-VM-Allocate 函数。但是,在实际部署中,当确定临界值时并不需要调用 ODAA-VM-Allocate 函数。对申请成功的用户 j ,需要以他们的非升次序考虑在时间 t 没有申请成功的用户 j' ,并且弄清如果 j 没有参与,用户 j' 能否申请成功。只需比较 j 占据的资源 and j' 请求的资源,就可以弄清这一点。当首次发现这一用户 j' 时,用户 j 的费用为 $\hat{v}_j$ 。需要在算法开始时对失败的申请进行一次排序,这一过程的复杂度仍然为 $O(n \log n)$ 。因此,ODAA-VM 的时间复杂度为 $O(n \log n)$ 。

定理 3 ODAA-VM 的竞争比为 η 。

设有两个申请 θ_1 和 θ_2 , 且 $s_1=1, s_2=M, (a_1, l_1, d_1, v_1)=(0, l, l, v)$,

$(a_2, l_2, d_2, v_2)=(1, l, l+1, Mv+\varepsilon), l>1$ 。一方面, ODAA-VM 将为用户 1 分配其在时间 $l>1$ 要释放的资源。因为对于 θ_2 , 有 $a_2+l_2=1+l=d_2$, 所以用户 2 必须在时间 $a_2=1$ 申请成功, 否则用户 2 需要收回申请。由于其无法在截止日期前完成作业, 因此 ODAA-VM 只会为用户 1 分配 VM 实例, 并获得价值 v (式(1))。另一方面, 最优离线算法将为用户 2 分配资源, 获得价值 $Mv+\varepsilon>M \times v$ 。因此, ODAA-VM 算法的竞争比为 η 。

6 实验结果

通过对 ODAA-VM 和一种高性能离线算法的比较, 证明 ODAA-VM 的优缺点。

6.1 实验配置

对参考文献 [8] 中的 CA-Provision 离线算法与本文的 ODAA-VM 进行比较。CA-Provision 算法解决了云环境下的动态 VM 供应和分配问题。以固定时间间隔(如 1 h)调用 CA-Provision 算法。当调用时, 该算法考虑上次间隔期间收集的所有申请, 通过组合拍卖来确定申请成功的用户、费用及分配的 VM 实例数量。分配的 VM 实例只能使用一个时间间隔, 用户如果继续访问资源, 就必须继续申请, 直到满足这些用户的时间要求。如果需求量上升, 则当前分配给用户的资源在以后可能被抢占。

本文使用的输入来自于参考文献[11]中的 Parallel Workload Archive(并行工作负载数据库)。表 1 给出了工作负载的日志信息及相关统计信息, 前 4 列分别为工作负载日志的名称、日志的采集时间、作业数量及日志总小时数, 第 5 列给出了生成日志的系统的处理器数量。假设 VM 实例的权重对应于分配给该实例的处理器数量, 则一个系统的处理器总数量表示计算资源的总权重 s , 将分配给工作负载 ω 的总体计算资源表示为 M_ω 。表 1 的后 4 列表示相关统计数据: J_ω 表示每小时递交给系统的平均作业数量; 平均运行时间 T_ω 及单个作业的平均处理器数量 P_ω 在计算

表 1 工作负载日志

日志文件名称	采集周期/月	作业总量/个	总小时数/个	处理器数/个	作业量	平均运行时间/h	每个作业的平均处理器数量/个	正规化负载
DAS2-fs0-2003	12	225 711	8 744	144	25.81	1.09	10.27	2.01
DAS2-fs1-2003	12	40 315	8 633	64	4.67	1.23	8.38	0.75
DAS2-fs2-2003	12	66 429	8 760	64	7.58	1.29	9.45	1.44
DAS2-fs3-2003	12	66 737	8 712	64	7.66	1.17	4.96	0.69
DAS2-fs4-2003	11	33 795	7 963	64	4.24	1.67	3.66	0.40
LLNL-Ubg1-2006	7	112 611	5 339	2 048	21.09	1.25	575.79	7.41
LPC-EGEE-2004	9	234 889	5 728	140	41.00	1.80	1	0.35

时考虑工作负载日志的所有记录;最后一列为负载 ω 的“正规化负载” η_ω ,其计算式为:

$$\eta_\omega = \frac{J_\omega \times T_\omega \times P_\omega}{s_\omega} \quad (10)$$

正规化负载用来衡量请求的平均资源量与可用的单位计算资源之间的相对关系,帮助对异构工作负载进行排序,并解释部分实验结果。

对仿真实验的设置如下。假设云供应商提供 4 种类型的 VM 实例资源 $VM_1, \dots, VM_4$, 且实例资源的权重 $\omega=(1,2,4,8)$ 。利用负载 ω 及上文给出的已知统计数据生成用户申请 $(\theta_j=r_1^j, \dots, r_m^j, a_j, l_j, d_j, v_j)$, 根据日志文件确定用户 j 的到达时间 a_j 。 $s_j = \sum_{i=1}^m \omega_i r_i^j$, 即被请求资源的总权重由均值为 $s_{avg}=P_\omega$ (负载 ω 每个作业的平均处理器数量) 的指数分布生成。确定 s_j 后随机确定 s_j 的值, 同时利用均值为 l_{avg} 的指数分布确定分配资源的使用时间, l_{avg} 由负载 ω 的平均作业时间 T_ω 确定。截止时间 $d_j = a_j + l_j \times \exp(d_{avg})$, 价值 $v_j = \exp(v_{avg})$ 。表 2 给出了 d_{avg} 和 v_{avg} 的值。通过随机选择 50% 的用户, 并将他们的所有申请参数乘以 2 来增加输入数据的多样性。

表 2 仿真参数

参数	标记法	数值
批量实例的平均权重	s_{avg}	基于工作负载(P_ω)
每个作业的平均时间	l_{avg}	基于工作负载(T_ω)
平均期限因子	d_{avg}	3
平均价值	v_{avg}	5、10、20
单个空闲 VM 的运行成本	c_R, c_1	1, 0.5

CA-Provision 算法根据单个空闲 VM 实例的相关成本参数计算底价 $v_{ns}=c_R-c_1$ 。虽然本文没有给出带有底价的 ODAA-VM, 但是通过抛弃低于底价的用户, 并对申请成功的用户收取底价费用而不是 0 费用 (ODAA-VM- Payment 函数的第 20 行), 即可集成底价策略。在本文实验中, ODAA-VM 的底价与 CA-Provision 算法的底价相同。表 2 给出了所有参数值, 对每个工作负载日志, 基于不同的参数组合进行 18 次实验。

6.2 结果分析

在图 1 中总结了每个工作负载的结果。水平轴表示工作负载和正规化负载, 垂直轴表示成功申请的用户比例、每个成功申请的用户的平均收入以及每个成功申请的用户的平均效用。在图 1(a)中, ODAA-VM 的用户服务比例高于 CA-Provision 算法。最明显的性能增益出现于日

志 LLNLuBGL-2006 (正规化负载 7.41) 和 DAS2-fs0-2003 (正规化负载 2.01) 中, 且是这里使用的工作负载的最大值。DAS2-fs0-2003 负载的用户服务比例几乎翻倍, LPC-EGEE-2004 负载的用户服务比例增长将近 6 倍。这是因为在 CA-Provision 算法中, 即使有可用资源, 用户也必须等待下次拍卖时间。那时, 可能有更多用户到达, 因此该用户可能拍卖失败。此外, 对于部分用户来说, 即使在部分拍卖中取得成功, 但是如果他们需要长期占用资源来完成任务, 将可能发生资源抢占, 所以他们会选择离开。由于上述两种工作负载增加了申请密度, 所以导致拍卖数量上升, 而 ODAA-VM 由于其在线设计, 可以容纳更多的申请。

从图 1(b)可知, 每个成功申请用户带来的平均收益下降。由于在线算法根据不完全信息确定分配策略, 所以无法实现收入最优。一种直观解释就是 ODAA-VM 通过尽快做出分配决策, 帮助用户避免与后续用户竞争。对于 CA-Provision 算法, 用户必须与相同周期内的其他用户展开竞争, 因此价格将会上升, 但是成功申请用户的数量上升弥补了收益损失。

从图 1(c)可以看出, ODAA-VM 成功申请用户的平均效用与 CA-Provision 算法相当。虽然本文希望通过降低用户支出提高用户的效用, 但是前提是两次拍卖所选择的分配用户集合相同。这是不可能实现的, 原因有两点: 一是 CA-Provision 算法会抢占价值较低的用户, 将资源分配给价值高的用户, 而 ODAA-VM 会把 VM 分配给用户, 并让用户在其所申请的整个时段内使用资源, 但是如果有价值更高的用户到达, 则这可能导致没有可用资源, 从而丢失该用户; 二是 ODAA-VM 中成功申请的用户数量要多于 CA-Provision 算法, 这说明 ODAA-VM 容纳了低价值用户, 而这些用户导致平均效用较低。

总体来说, 本文认为 ODAA-VM 提升了用户和云供应商的云体验, 云供应商服务了更多用户, 增加了用户满意度, 最终增加了总体收入。

7 结束语

本文设计了一种云环境下 VM 在线供应和分配算法。在该算法下, 每当有足够多的资源和合适的申请, 就会分配 VM 实例。本文算法对用户具有激励作用, 且可在多项式时间内运行; 可增加被服务的用户数量, 但是降低了平均收益, 然而增加的用户数量可以弥补平

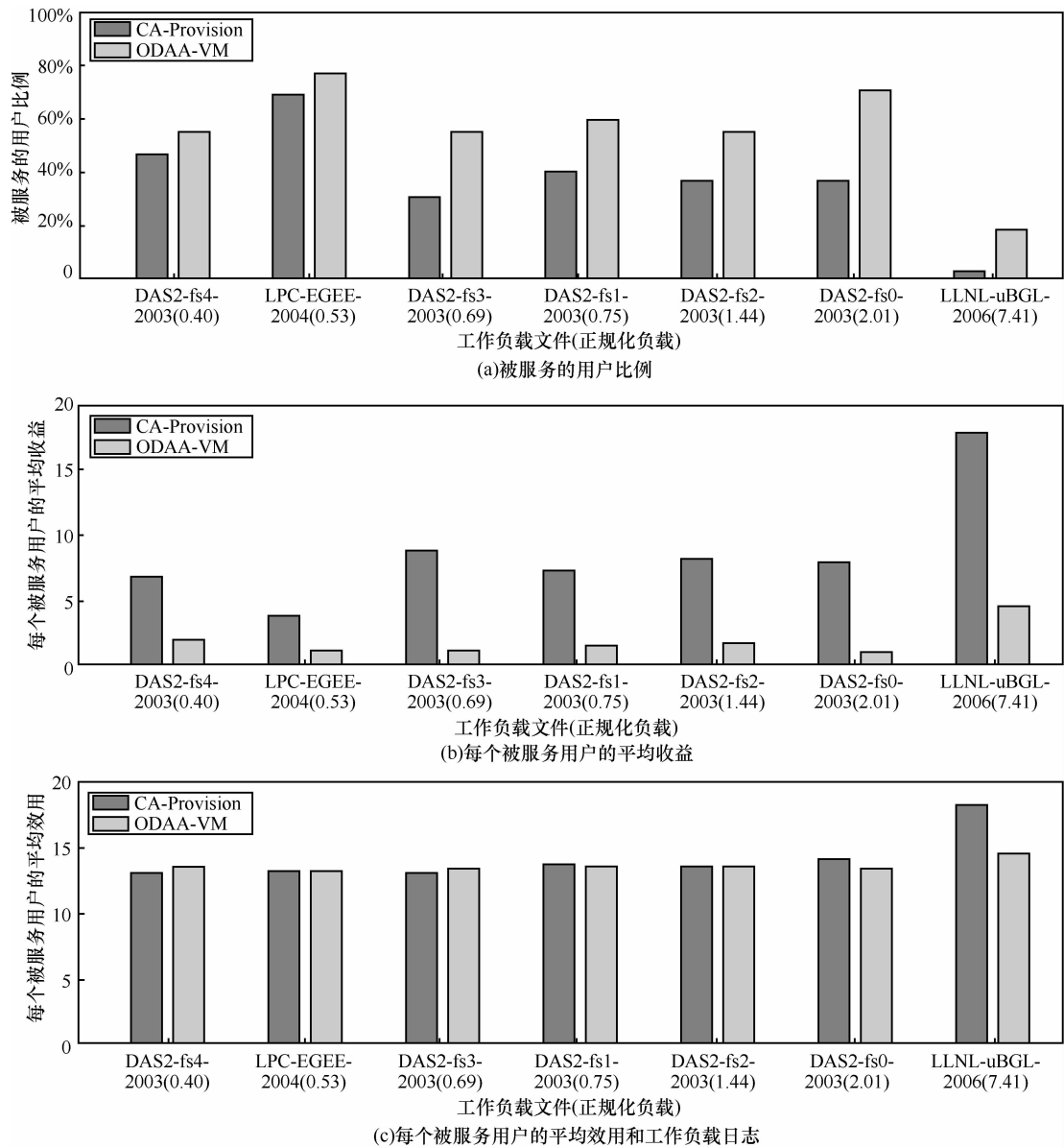


图1 CA-Provision 算法和 ODAA-VM 的总体结果

均收益损失。在未来工作中,将针对 CA-Provision 算法中用户 VM 被抢占的情况,设置相应的赔偿代价,并将其再次与 ODAA-VM 进行比较,观察“用户的平均收益”是否有所改进。

参考文献

- 1 刘正伟, 文中领, 张海涛. 云计算和云数据管理技术. 计算机研究与发展, 2012, 49(Z1): 26~31
Liu Z W, Wen Z L, Zhang H T. Cloud computing and cloud data management technology. Journal of Computer Research and Development, 2012, 49(Z1): 26~31
- 2 范杰, 彭舰, 黎红友. 基于蚁群算法的云计算需求弹性算法.

计算机应用, 2011, 31(1): 1~4

- Fan J, Peng J, Li H Y. Demand elasticity algorithm for cloud computing based on ant colony optimization algorithm. Journal of Computer Applications, 2011, 31(1): 1~4
- 3 Lin W Y, Lin G Y, Wei H Y. Dynamic auction mechanism for cloud resource allocation. Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), Melbourne, Australia, 2010: 591~592
- 4 Parkes D C, Singh S. An mdp-based approach to online mechanism design. Proceedings of the 17th Annual Conference on Neural Information Processing Systems, Sydney, Australia, 2003:1421~1426

(2015048-9 接排在 2015088-6 后)



IEEE Transactions on Signal Processing, 2011, 59(1): 399~414

- 7 Zhang H, Lin Y, Tao M X. Resource allocation with subcarrier pairing in OFDM two-way relay networks. IEEE Wireless Communications Letters, 2012, 1(2): 61~64
- 8 Alam M S, Mark J W, Shen X M. Relay selection and resource allocation for multi-user cooperative OFDMA networks. IEEE Transactions on Wireless Communications, 2013, 12(5): 2193~2205
- 9 Li T, Yu H, Liu S. Adaptive power allocation for decode-and-forward MIMO-OFDM relay system. Proceedings of 5th International Conference on Wireless Communications, Beijing, China, 2009: 1~4
- 10 Zu L J, Ji Y S, Wang L P, *et al.* Joint optimization in multi-user MIMO-OFDMA relay-enhanced cellular networks. Proceedings of Wireless Communications and Networking Conference, Istanbul, Turkey, 2011: 13~18
- 11 Lin Y B, Wu W H, Su Y T. Optimal and suboptimal resource allocation for multi-hop MIMO-OFDMA networks. Proceedings of Personal Indoor and Mobile Radio Communications, Sydney, Australia, 2012: 502~506
- 12 张冬梅, 徐友云, 蔡跃明. OFDMA 系统中线性注水功率分配算法. 电子与信息学报, 2007, 29(6): 1286~1289
Zhang D M, Xu Y Y, Cai Y M. Linear water-filling power allocation in OFDMA system. Journal of Electronics & Information Technology, 2007, 29(6): 1286~1289

2015088-6

(上接 2015048-8)

- 5 Carroll T E, Grosu D. Incentive-compatible online scheduling of malleable parallel jobs with individual deadlines. Proceedings of the 39th International Conference on Parallel Processing, San Diego, CA, USA, 2011: 418~425
- 6 师雪霖, 徐格. 云虚拟机资源分配的效用最大化模型. 计算机学报, 2013, 36(2): 252~262
Shi X L, Xu K. Utility maximization model of virtual machine scheduling in cloud environment. Chinese Journal of Computers, 2013, 36(2): 252~262
- 7 罗贺, 孙锦波, 胡笑旋等. 云商务环境下的多源信息服务资源分配模型. 计算机集成制造系统, 2013, 19(10): 137~142
Luo H, Sun J B, Hu X X, *et al.* Resource allocation model for multi-sources information service in cloud business. Computer Integrated Manufacturing Systems, 2013, 19(10): 137~142
- 8 Zaman S, Grosu D. Combinatorial auction-based dynamic vm provisioning and allocation in clouds. Proceedings of IEEE 3rd International Conference on Cloud Computing Technology and Science (CloudCom), Athens, Greece, 2011: 107~114
- 9 Zaman S, Grosu D. A combinatorial auction-based mechanism for

[作者简介]



郭涛,男,宁波大学通信技术研究所以硕士生,主要研究方向为中继通信系统中的资源分配等。



李有明,男,宁波大学通信技术研究所以教授,主要研究方向为无线和有线宽带通信技术,包括干扰抑制、动态资源分配、MIMO-OFDM、协作中继等。

雷鹏,男,宁波大学通信技术研究所以硕士生,主要研究方向为认知无线电中的频谱感知与资源分配等。

季彪,男,宁波大学通信技术研究所以硕士生,主要研究方向为水声通信中的多普勒补偿技术等。

李程程,男,宁波大学通信技术研究所以硕士生,主要研究方向为水声通信信道估计等。

dynamic VM provisioning and allocation in clouds. IEEE Transactions on Cloud Computing, 2013,1(2): 129~141

- 10 Mu'alem A, Nisan N. Truthful approximation mechanisms for restricted combinatorial auctions. Proceedings of the 18th National Conference on Artificial Intelligence, Edmonton, Alberta, Canada, 2002: 379~384
- 11 Lublin U, Feitelson D G. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. Journal of Parallel and Distributed Computing, 2003, 63(11): 1105~1122

[作者简介]



张丽敏,女,西安外事学院现代教育技术中心讲师,主要研究方向为云计算、网络安全、计算机仿真。

2015048-9