基于动态规划的虚拟机放置策略

张 勋[1] 顾春华[1] 罗 飞[1] 常耀辉[1,2] 文 赓[3]

(华东理工大学信息科学与工程学院 上海 200237)[1] (石河子大学信息科学与技术学院 石河子 832003)[2]

(上海电力学院计算机科学与工程学院 上海 200090)[3]

摘 要 在IaaS云环境中,资源的分配管理关键取决于如何放置虚拟机,不当的放置策略可造成资源的损耗以及更多的能耗开销。为了降低整个数据中心的资源损耗和能耗开销,建立一个多目标优化的问题模型,并提出了一种基于动态规划思想的虚拟机放置策略。策略将放置问题转化为多阶段决策的背包问题,利用动态规划的思想把背包问题划分成一系列规模更小的子问题,通过求解子问题的最优解得到原问题的最优解。仿真实验表明,该策略能大大降低数据中心的能耗,并减少资源损耗。

关键词 云环境,资源损耗,能耗开销,多目标优化,动态规划

中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.08.010

Virtual Machine Placement Strategy Based on Dynamic Programming

ZHANG Xun[1] GU Chun-hua[1] LUO Fei[1] CHANG Yao-hui[1,2] WEN Geng[3]

(School of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237, China)[1]

(College of Information Science and Technology, Shihezi University, Shihezi 832003, China)[2]

(School of Computer Science and Engineering, Shanghai University of Electric Power, Shanghai 200090, China)[3]

Abstract In the environment of IaaS cloud, the key factor of the resource allocation management is how to place virtual machine. It is mostly probably that an improper placement strategy may cause the loss of resource and more energy consumption. Thus, a multi-objective optimization model was established which devotes to reduce the resource loss and energy consumption of the whole data center. Further more, a kind of placement strategy about virtual machine based on dynamic programming theory was proposed. In the strategy, the problem of placement is transformed into a knapsack problem of multi-stage decision, in which the knapsack problem is divided into a series of smaller sub-problems with the idea of dynamic programming. And the optimal solution of the original problem is obtained through solving the optimal solution of the sub-problems. Finally, the simulation experiment shows that this strategy can greatly reduce the energy consumption and the resource loss of data center.

Keywords Cloud environment, Resource loss, Energy consumption, Multi-objective optimization, Dynamic programming

1 引言

云计算[1]作为一种新兴的计算模式,为我们带来了巨大的经济效益;同时,它的快速发展也使得数据中心的规模不断扩大,巨大的能耗问题随之而来[2],世界范围内数据中心的年均能耗量在不断增长[3]。另外,如果数据中心的资源分配不合理,产生的资源损耗不仅会带来更多的资源成本,也会增加整个数据中心的能耗。

数据中心由一系列主机组成,若主机上各维资源的使用不均,则会造成整个系统的资源损耗,其能耗开销也是整个系统能耗的关键组成部分。虚拟机是系统进行资源分配的基本单位,虚拟机的放置结果将决定整个系统的资源分配状况,进而影响着整个系统的资源损耗和能耗开销。因此,为了降低整个系统的资源损耗和能耗开销,研究虚拟机的放置问题[4],优化虚拟机与物理主机的映射关系,提高系统资源分配的合理性,是十分必要的。

本文以最小化能耗和资源损耗为优化目标,从宿主机选择、虚拟机排序和规划放置3个方面进行考虑,在规划放置时将放置问题转化为背包问题,结合动态规划解决多阶段决策最优化问题的思想方法,最终提出一种基于动态规划[5]的虚拟机放置策略(Dynamic Programming for Virtual Machine Placement, DP_VMP)。仿真实验的结果表明,相比于传统的

到稿日期:2016-07-19 返修日期:2016-11-07 本文受国家自然科学基金项目(61472139)资助。

张 勋(1991—),男,硕士生,主要研究方向为云计算,E-mail:zx_zhangxun@163.com;顾春华(1970—),男,博士,教授,博士生导师,主要研究方向为物联网、云计算、软件工程;罗 飞(1978—),男,博士,副教授,硕士生导师,主要研究方向为云计算、嵌入式系统;常耀辉(1981—),男,硕士生,讲师,主要研究方向为云计算、数据挖掘;文 赓(1992—),男,硕士生,主要研究方向为嵌入式系统、物联网。

启发式虚拟机放置策略，该策略能使整个数据中心的能耗及资源损耗变得更低。

2　相关工作

在 IaaS 云环境下，将 N 个虚拟机放置到 M ($M \leqslant N$) 个物理主机上，其解空间为 M^N，这是一个典型的 NP-hard 问题[6-7]，几乎不可能在多项式时间内找到最优解（除非 P=NP）。该类问题的研究重点是寻求近似算法，试图找到一个近似最优解。

针对该放置问题，传统的解决方案是基于贪心策略的启发式算法，如最佳适应（Best Fit，BF）、首次适应（First Fit，FF）以及降序最佳适应（Best Fit Decreasing，BFD）和降序首次适应（First Fit Decreasing，FFD）等。文献[8]通过将数据中心的节点整合描述为随机装箱优化问题来处理负载的随机性，但是只考虑了以 CPU 资源作为约束条件；文献[9]首先计算各物理主机的目标函数值，然后将虚拟机部署在函数值最优的主机上，但是没有考虑资源利用率的问题。由此看来，启发式算法通常是单点搜索，很少对多个目标同时优化，并且很容易陷入局部最优解。

近年来，一些元启发式算法被用来求解该问题。文献[10]考虑了物理机的资源损耗和网络总流量两个目标，提出一种多目标蚁群优化的虚拟机放置算法，仿真实验结果表明该算法比 FFD 算法更加有效；文献[11]考虑了物理主机的资源浪费和数据中心的网络时延两个目标，利用改进型的双适应度遗传算法（CGA）进行多目标的优化，最后通过与贪心算法的比较证明了 CGA 的有效性。但是，元启发式算法通常存在参数较多、复杂度较高、不易实现等缺点。

综上所述，虚拟机放置问题不仅要考虑对系统能耗的影响，还要考虑各维度资源的利用情况，另外还要确保放置算法的可实现性。本文综合了系统 CPU 和内存两个维度的资源使用情况，提出使用动态规划的思想解决虚拟机的放置问题，使得数据中心的系统能耗和资源损耗最小。

3　问题建模

云平台虚拟化的本质就是对数据中心的物理资源（计算、存储和网络）进行抽象，从而形成一个可扩展、可按需分配的虚拟资源池。如图 1 所示的应用场景中，用户集合 $UserList=(User_1, User_2, \cdots, User_i)$ 向云平台提交了 n 个申请虚拟机的请求，请求集合为 $VMList=(VM_1, VM_2, \cdots, VM_n)$，这些虚拟机将被放置在 m 个物理主机上，物理主机集合为 $PHList=(PH_1, PH_2, \cdots, PH_m)$。在处理这些请求时，我们把如何选择合适的物理宿主机称为虚拟机放置问题。

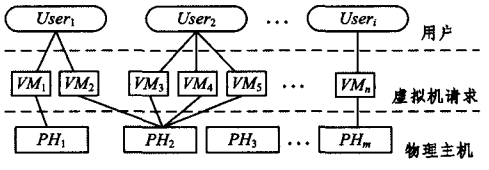

图 1　虚拟机放置问题的场景图

在上述放置场景中，虚拟机的规格通常由云服务提供者定制，其规格套餐也是有限的，并且数据中心的物理主机往往是异构的。本文的主要目标是在虚拟机与物理主机之间的映射关系上进行优化，使得放置完虚拟机后，系统的总能耗减少，资源损耗降低。下面分别对系统的能耗和资源损耗进行建模，并根据这两个模型建立一个虚拟机放置的多目标约束优化模型。

3.1　能耗模型

随着云计算的发展，数据中心不断扩大，高能耗的问题也变得更加突出，优化虚拟机放置策略可使数据中心绿色节能，并能有效降低运营成本。数据中心的能耗 E 通常由多部分构成（主机集群的能耗以及空调、照明等附属设备的能耗），由于附属设备的能耗 E_{other} 与其工作时间近似成正比，因此本文主要关注主机集群的能耗 $E_{cluster}$。设第 j 台物理主机的功耗为 P_j，则在工作时间 T 内主机集群的能耗 $E_{cluster}$ 为：

$$E_{cluster}(T)=\int_0^T \left(\sum_{j=1}^{m} P_j\right) \mathrm{d}t$$

降低主机集群能耗的关键在于降低主机集群的总功耗。相关文献[12]的研究表明，物理主机的功耗与其 CPU 利用率呈近似线性关系。文献[13]通过长达 3 个月的监控实验得出，主机的功耗与其 CPU 利用率的相关系数高达 0.990667。此外，文献[14]还显示，物理主机 CPU 即使处于空闲状态，也将消耗在满负荷时 70%左右的电力。

根据上述功耗特征，定义单个物理主机的功耗为：

$$P_j=\begin{cases}(P_j^{max}-P_j^{idle})\times U_j^{cpu}+P_j^{idle}, & 0\leqslant U_j^{cpu}\leqslant 1\\ 0, & \text{otherwise}\end{cases} \tag{1}$$

其中，U_j^{cpu} 表示第 j 个物理主机 CPU 的利用率，$j\in[1,2,\cdots,m]$；P_j^{max} 和 P_j^{idle} 分别表示第 j 个物理主机满负荷和空闲时的功耗，即 CPU 利用率为 100%与 0%时的功耗；P_j 表示第 j 个物理主机 CPU 利用率为 U_j^{cpu} 时的功耗。则整个数据中心的总功耗为：

$$P=\sum_{j=1}^{m} P_j \tag{2}$$

能耗优化目标函数为：

$$\begin{aligned}\min P &= \min\sum_{j=1}^{m} P_j\\ &= \min\sum_{j=1}^{m}[y_j\times((P_j^{max}-P_j^{idle})\times\sum_{i=1}^{n}(x_{ij}\cdot D_{ij}^{cpu})+\\ &\quad P_j^{idle})]\end{aligned} \tag{3}$$

其中，y_j 表示第 j 个物理主机是否正在使用，如果正在使用，则 $y_j=1$，否则 $y_j=0$；x_{ij} 表示第 i 个虚拟机是否放置到第 j 个物理主机上，如果是，则 $x_{ij}=1$，否则 $x_{ij}=0$；D_{ij}^{cpu} 表示第 i 个虚拟机和第 j 个物理主机上的 CPU 所占比。

本模型中分配给虚拟机的资源都是指标准化后的虚拟资源，物理主机的资源利用率设定为已分配的虚拟资源和物理主机虚拟资源总量的比值。虚拟机部署在物理主机上，由于其资源利用具有动态性变化的特点，为保证云平台服务质量 QoS 并满足用户服务水平协议 SLA，在实验中不考虑虚拟机放置后其自身资源利用情况对物理主机利用率的影响，以实际请求的标准化资源作为其分配得到的资源数量。

3.2　资源损耗模型

当一个物理主机上各维资源的利用率不均衡时，就称该主机上存在着资源损耗，损耗的大小标志着各维资源利用情

况的不均衡程度。由于物理主机上往往运行着多种规格的虚拟机，不同规格的虚拟机对物理主机各维资源的占用率也不同，如果放置不合理，则可能使物理主机各维资源利用率存在较大差异，也即存在较大的资源损耗。如图 2 所示，该主机上已放置了 3 个不同规格的虚拟机，剩余 40%的 CPU 和 10%的内存，此时用户又有一个占 CPU 10%、占内存 20%的虚拟机请求，但因为主机的内存资源不足而无法放置该虚拟机，所以又将开启一台新的主机来放置该虚拟机，这就造成了资源的损耗以及更大的系统能耗。

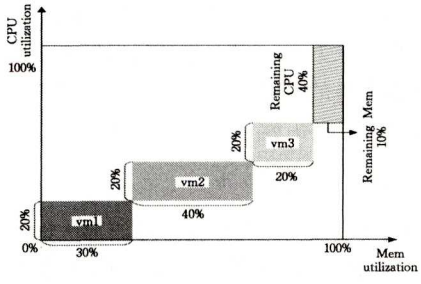

图 2 资源损耗示例图

物理主机所具备的资源维度与虚拟机所请求的资源维度是一致的，假设都是 r 维资源，R_{jk} 为第 j 个物理主机第 k 维资源的剩余率，其中 $k\in[1,2,\cdots,r]$。则第 j 个物理主机的资源损耗表示为：

$$W_j=\sum_{k=1}^{r}(R_{jk}-\min\{R_{j1},R_{j2},\cdots,R_{jr}\}) \tag{4}$$

本文由于只考虑 CPU 和内存两维资源，因此可将第 j 个物理主机的资源损耗表示为：

$$W_j=|R_j^{cpu}-R_j^{mem}| \tag{5}$$

其中，R_j^{cpu} 表示第 j 个物理主机 CPU 的剩余率，R_j^{mem} 表示第 j 个物理主机内存的剩余率。

资源损耗表示各维资源利用率的均衡程度，并不能完全反映资源的利用情况。为了提高这个均衡度，即在保证资源损耗较小的同时提高资源的利用率，将资源利用率模型加入资源损耗模型中，得到第 j 个物理主机最终的资源损耗模型为：

$$W_j=-|R_j^{cpu}-R_j^{mem}|\times\ln(\max\{U_j^{cpu},U_j^{mem}\}) \tag{6}$$

则整个数据中心的总损耗为：

$$W=\sum_{j=1}^{m}W_j \tag{7}$$

由于 $R_j^{cpu}+U_j^{cpu}=1$，$R_j^{mem}+U_j^{mem}=1$，因此损耗目标函数可以表示为：

$$\begin{aligned}\min W=&\min\sum_{j=1}^{m}[y_j\times(-|(1-U_j^{cpu})-(1-U_j^{mem})|\times\\&\ln(\max\{U_j^{cpu},U_j^{mem}\}))]\\=&\min\sum_{j=1}^{m}[y_j\times(-|\sum_{i=1}^{n}(x_{ij}\cdot D_{ij}^{cpu})-\sum_{i=1}^{n}(x_{ij}\cdot\\&D_{ij}^{mem})|\times\ln(\max\{\sum_{i=1}^{n}(x_{ij}\cdot D_{ij}^{cpu}),\sum_{i=1}^{n}(x_{ij}\cdot\\&D_{ij}^{mem})\}))]\end{aligned} \tag{8}$$

其中，D_{ij}^{mem} 表示第 i 个虚拟机与第 j 个物理主机的内存所占比。

3.3 虚拟机放置的多目标约束优化模型

基于 3.1 节和 3.2 节中的两个目标函数，再加入适当的约束条件即可得到虚拟机放置的多目标约束优化模型：

$$\min P \text{ and } \min W \tag{9}$$

Subject to：

$$\begin{cases}\sum_{j=1}^{m}x_{ij}=1,i\in[1,2,\cdots,n] & (10)\\ \sum_{i=1}^{n}x_{ij}\cdot D_{ij}^{cpu}\leqslant C_j^{cpu}\cdot y_j,j\in[1,2,\cdots,m] & (11)\\ \sum_{i=1}^{n}x_{ij}\cdot D_{ij}^{mem}\leqslant C_j^{m}\cdot y_j,j\in[1,2,\cdots,m] & (12)\\ x_{ij}\in\{0,1\},i\in[1,2,\cdots,n],j\in[1,2,\cdots,m] & (13)\\ y_j=\begin{cases}0, & \text{if } \sum_{i=1}^{n}x_{ij}=0\\ 1, & \text{if } \sum_{i=1}^{n}x_{ij}>0\end{cases},j\in[1,2,\cdots,m] & (14)\end{cases}$$

式(10)表示任意一个虚拟机只能放置在一台物理宿主机上；式(11)、式(12)分别表示放置在某一物理主机上的所有虚拟机的 CPU 和内存资源需求必须在该主机的资源总量约束下；式(13)、式(14)表示 x_{ij} 和 y_j 取 0 或 1。

4 虚拟机放置策略 DP_VMP

根据前文建立的多目标优化模型，设计虚拟机的放置策略。虚拟机的放置问题可以被看成是多维的装箱问题。本文将多维的装箱问题转化为多维的 0-1 背包问题，每次只考虑对单个背包的装入。当背包无法装入全部的物品时，再考虑使用新的背包。DP_VMP 放置策略主要分为 3 个步骤：宿主机选择、虚拟机排序和规划放置。

4.1 宿主机选择

在为虚拟机选择宿主机时，为了减少系统能耗，总是希望虚拟机的放置对系统所造成的能耗影响最小。为了选出合适的宿主机，首先根据物理主机的资源情况对其放置能力进行评价排序，定义评价函数为：

$$f_j=f(PH_{j1},PH_{j2},\cdots,PH_{jr}) \tag{15}$$

其中，$f(PH_{j1},PH_{j2},\cdots,PH_{jr})$ 根据第 j 台物理主机的各维度的资源情况对其进行能力评价，返回值 f_j 表示该主机的能力值。假设系统为用户提供的虚拟机套餐集合为 $F=(F_1,F_2,\cdots,F_t)$，其中套餐 $F_l=(F_{l1},F_{l2},\cdots,F_{lr})$，$F_{lk}$ 为套餐 l 对第 k 维资源的需求量，所有套餐对第 k 维资源的平均需求量为 $a_k=\frac{(\sum_{l=1}^{t}F_{lk})}{t}$。本文只考虑 CPU 和内存两维资源。由于主机的功耗和其 CPU 利用率呈近似线性关系，因此引入 CPU 所占的权重系数 α，则放置能力评价函数可表示为：

$$f_j=\begin{cases}\frac{PH_j^{mem}}{a^{mem}}+\alpha(\frac{PH_j^{cpu}}{a^{cpu}}-\frac{PH_j^{mem}}{a^{mem}}), & \frac{PH_j^{mem}}{a^{mem}}<\frac{PH_j^{cpu}}{a^{cpu}}\\ \frac{PH_j^{cpu}}{a^{cpu}}, & \text{otherwise}\end{cases} \tag{16}$$

其中，PH_j^{cpu} 和 PH_j^{mem} 分别表示第 j 台主机的 CPU 总量和内存总量，a^{cpu} 和 a^{mem} 分别表示所提供套餐的 CPU 平均需求量和内存平均需求量。

数据中心的物理主机通常是异构的，其能耗往往也是不同的，为了节省整个数据中心的系统能耗，本文优先将虚拟机

放置在单位能耗下放置能力最强的物理主机上，故可将单位能耗下的放置能力表示为：

$$fp_j=\frac{f_j}{P_j^{max}} \tag{17}$$

4.2 虚拟机排序

在对虚拟机请求集合进行单节点的放置时，为保证虚拟机资源分配的灵活性，本文考虑优先把资源分配给需求较大的虚拟机，因此需要对虚拟机的资源需求进行排序。由于用户的虚拟机配置取决于用户所选择的虚拟机套餐，因此先对套餐的资源需求进行排序，定义第 l 个套餐在第 j 台物理主机上的需求程度函数为：

$$n_{lj}=\sum_{k=1}^{r}\frac{F_{lk}}{PH_{jk}} \tag{18}$$

基于 CPU 和内存两维资源的需求程度函数可表示为：

$$n_{lj}=\frac{F_l^{cpu}}{PH_j^{cpu}}+\frac{F_l^{mem}}{PH_j^{mem}} \tag{19}$$

其中，F_l^{cpu} 和 F_l^{mem} 分别表示第 l 个套餐的 CPU 需求量和内存需求量。

4.3 规划放置

放置思想描述：将一批虚拟机请求放置在数据中心的物理主机上，首先选出单位能耗下放置能力最强的物理主机，当该主机的剩余资源能够满足请求的总资源需求时，将请求的虚拟机放置在该主机上；若其不能满足资源需求，则对请求集合中的虚拟机进行排序，并基于动态规划的思想把放置问题转换为背包问题，将宿主机当作一个背包，背包有二维资源 PH_j^{cpu} 和 PH_j^{mem}，每一个请求的虚拟机可看作是一个要放入背包的物品，最后选出合适的物品放入背包，使得各维资源的利用率平衡且较高。若放置处理结束后仍有虚拟机请求剩余，则再从剩余主机中选出单位能耗放置能力最强的主机重复以上的放置过程，直至所有虚拟机请求处理完毕或系统资源使用完毕。

基于上述放置思想，DP_VMP 放置策略可用算法 1 描述如下。

算法 1　基于动态规划的虚拟机放置策略 DP_VMP

输入：虚拟机请求集合 VMList，可用主机集合 PHList，CPU 所占用的权重系数 α(本文取 0.6)

输出：虚拟机放置结果集合 MapSet

1. 初始化结果集合 MapSet 为空集合
2. While VMList is not empty
3. If PHList is empty
4. 通知管理员添加资源，break 跳出循环
5. EndIf
6. 按式(17)降序排列 PHList，Set PHost=PHList.get(0)
7. Calculate the total cpu and memory requirements of VMList
8. If PHost can meet the requirements of VMList resources
9. Add assignment 〈VMList，PHost . id〉 to MapSet
10. VMList.clear()
11. Else
12. 按式(19)降序排列 VMList
//对 PHost 使用单节点的规划放置策略 DP_SN_VMP
13. Set MapSet=DP_SN_VMP(VMList，PHost，α，MapSet)
14. EndIf
15. PHList.remove(PHost)
16. EndWhile
17. Return MapSet

在对虚拟机进行单节点的规划放置时，首先对 CPU 和内存资源进行递增遍历，根据每次遍历所给定的 CPU 和内存资源量，又将放置过程分为 N 个相互联系的阶段，其中第 i 个阶段代表在当前给定的资源下前 $i(i\leqslant N)$ 台虚拟机的放置情况。假设第 i 个阶段的状态变量有 $f_{cpu}[i][v^{cpu}][v^{mem}]$ 和 $f_{mem}[i][v^{cpu}][v^{mem}]$，它们分别表示当前阶段给定 CPU 总量为 v^{cpu}、内存总量为 v^{mem} 时放置处理前 i 台虚拟机所消耗的 CPU 和内存资源量，那么第 $i+1$ 个阶段的状态就取决于决策是否放置第 $i+1$ 台虚拟机。

如果第 $i+1$ 台虚拟机对资源的需求量超过了当前给定的资源总量，那么不能放置第 $i+1$ 台虚拟机，否则就做出放置的决策。如图 3 所示，本文引入两个欧氏距离以决策是否放置第 $i+1$ 台虚拟机，其中点 $V(V^x,V^y)$ 的坐标表示当前给定的 CPU 总量 v^{cpu} 和内存总量 v^{mem} 全被分配使用时该主机的 CPU 利用率和内存利用率；点 $S_i(S_i^x,S_i^y)$ 的坐标表示在当前给定的资源总量下放置处理前 i 台虚拟机后该主机的 CPU 利用率和内存利用率；点 $T_{i+1}(T_{i+1}^x,T_{i+1}^y)$ 的坐标表示做出放置第 $i+1$ 台虚拟机的决策后该主机的 CPU 利用率和内存利用率。

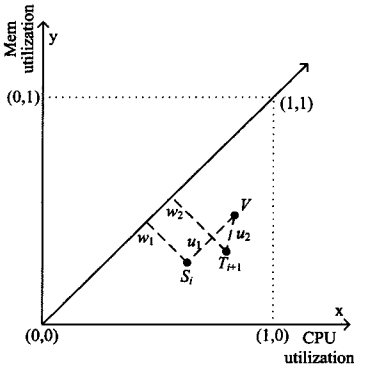

图 3　虚拟机放置决策图

点 S_i 与点 T_{i+1} 到点 V 的距离代表当前给定资源总量下该阶段的放置结果与资源被完全利用时的接近程度，如果 u_2 小于 u_1 则意味着做出放置第 $i+1$ 台虚拟机的决策能使该主机的资源利用更加充分。

点 S_i 与点 T_{i+1} 到直线 $y=x$ 的距离代表当前给定资源总量下该阶段放置结果的资源损耗情况。由于本文在资源损耗的目标函数中考虑了对资源的利用率，因此如果 $(-w_2\times\ln(\max\{T_{i+1}^x,T_{i+1}^y\}))$ 小于 $(-w_1\times\ln(\max\{S_i^x,S_i^y\}))$，则意味着做出放置第 $i+1$ 台虚拟机的决策能使该主机的资源损耗降低。

因此，做出放置第 $i+1$ 台虚拟机的决策条件是：

$$(u_2<u_1)\ \text{or}\ ((-w_2\times\ln(\max\{T_{i+1}^x,T_{i+1}^y\}))<(-w_1\times\ln(\max\{S_i^x,S_i^y\}))) \tag{20}$$

宿主机的 CPU 和内存的状态转移方程可表示为：

$$f_{cpu}[i+1][v^{cpu}][v^{mem}]=\begin{cases}f_{cpu}[i][v^{cpu}-VM_{i+1}^{cpu}][v^{mem}-VM_{i+1}^{mem}]+VM_{i+1}^{cpu}, & \text{如果式(20)满足}\\ f_{cpu}[i][v^{cpu}][v^{mem}], & \text{otherwise}\end{cases} \tag{21}$$

$$f_{mem}[i+1][v^{cpu}][v^{mem}] = \begin{cases} f_{mem}[i][v^{cpu}-VM_{i+1}^{cpu}][v^{mem}-VM_{i+1}^{mem}]+VM_{i+1}^{mem}, & \text{如果式(20)满足} \\ f_{mem}[i][v^{cpu}][v^{mem}], & \text{otherwise} \end{cases} \tag{22}$$

其中，VM_{i+1}^{cpu} 和 VM_{i+1}^{mem} 分别表示第 $i+1$ 台虚拟机的 CPU 和内存需求量。

待规划放置结束后，根据式(20)的决策条件可逆序求解出放置在该主机上的虚拟机集合。单节点(single node)动态规划放置策略 DP_SN_VMP 的伪代码如算法 2 所示。

算法 2 单节点的虚拟机规划放置策略 DP_SN_VMP

输入：虚拟机请求集合 VMList，物理宿主机 PHost，CPU 所占权重系数 α(本文取 0.6)，本次规划放置前的结果集 MapSet

输出：本次规划放置后的结果集 MapSet

```
1. Set n=VMList.size
2. For j=0:PHost.RemainCpu do
3.   For m=0:PHost.RemainMem do
4.     For i=0:n do
5.       If i==0 or j==0 or m==0
6.         Set f_cpu[i][j][m]=0 and f_mem[i][j][m]=0
7.       Else
8.         If j<VM_i^cpu or m<VM_i^mem
9.           Set f_cpu[i][j][m]=f_cpu[i-1][j][m]
10.          Set f_mem[i][j][m]=f_mem[i-1][j][m]
11.          Else
12.            Calculate these distances w1, w2, u1, u2
13.            Set the value of f_cpu[i][j][m] by formulas (21)
14.            Set the value of f_mem[i][j][m] by formulas (22)
15.          EndIf
16.        EndIf
17.    EndFor
18.  EndFor
19. EndFor
20. Set v^cpu=PHost.RemainCpu and v^mem=PHost.RemainMem
21. For i=n:1 do
22.   Calculate these distances w1, w2, u1, u2
23.   If meet the formula (20)
24.     Add assignment (VMi.id, PHost.id) to MapSet
25.     VMList.remove(VMi)
26.     Set v^cpu=v^cpu-VM_i^cpu and v^mem=v^mem-VM_i^mem
27.   EndIf
28. EndFor
29. Return MapSet
```

5 实验仿真与分析

5.1 仿真环境

本节采用 Java 语言对算法进行了实现，通过与其他经典算法进行对比分析来验证 DP_VMP 放置策略在系统能耗和资源损耗方面的有效性。实验模拟了一个包含有 50 台异构主机的云平台，其中有 20 台主机的配置为 4 核 CPU、70GB 内存，还有 20 台的配置为 6 核 CPU、100GB 内存，最后 10 台的配置为 8 核 CPU、140GB 内存。参考 SPEC 对各厂家服务器性能与能耗比的测试结果为各类型主机设置能耗参数；参考 OpenStack 云平台对物理资源进行超额分配，分别设置内存和 CPU 的超额分配系数为 1.5 和 16，并预留系统资源的 10%来保证其他服务的质量。用于虚拟机放置的资源信息如表 1 所列。参照使用 OpenStack 云平台所提供的默认 flavor 套餐，并为各 flavor 套餐设定使用的概率系数，套餐信息如表 2 所列。最终与经典对比算法进行比较，对比算法包括随机放置算法(Random)、首次适应算法(FF)、CloudSim 实现的 VmAllocationSimple 算法(VAS)以及 OpenStack 默认的最大内存剩余算法(MM)。

表 1 云平台资源信息

Number of hosts	CPU/Cores	Memory/GB	MaxPower/Watts	ActiveIdle/Watts	Ratio
20	57	95	250	175	0.7
20	86	135	300	210	0.7
10	115	189	400	280	0.7

表 2 虚拟机套餐信息

flavor	CPU/Cores	Memory/GB	Probability
m1.small	1	1	0.1
m1.medium	2	4	0.3
m1.large	4	8	0.15
m2.small	2	2	0.15
m2.medium	4	4	0.2
m1.xlarge	8	16	0.1

5.2 实验结果分析

本文以系统能耗和资源损耗作为评价指标，在虚拟机请求数量相同的情况下比较各放置策略的能耗和资源损耗，每组对比实验做 10 次并求其平均值作为最终结果。不同放置策略的放置结果如表 3 所列。

表 3 不同放置策略的放置结果

虚拟机数目	DP_VMP 能耗/损耗	Random 能耗/损耗	VAS 能耗/损耗	MM 能耗/损耗	FF 能耗/损耗
50	723.41/0.0732	6892.86/1.4264	2963.41/0.4242	2963.41/0.3908	748.55/0.1269
100	1200.56/0.0624	9622.75/2.048	4098.6/0.6606	3132.56/0.3714	1504.34/0.2234
150	1844.97/0.1058	10560.35/2.3037	7501.37/1.4928	7501.28/1.2993	2206.71/0.3575
200	2383.34/0.0928	11104.87/2.5656	7676.07/1.4477	7675.98/1.4566	3004.08/0.4598
250	2968.83/0.1576	11338.18/2.5696	7841.88/1.5827	7841.81/1.5166	3667.76/0.5936
300	3490.4/0.1689	11576.98/2.7573	7999.76/1.465	7999.67/1.5137	4321.45/0.7159
350	3968.7/0.1315	11789.89/2.7045	8170.38/1.5061	8328.52/1.461	5076.75/0.8084
400	4558.35/0.1855	11984.53/2.8053	11435.39/2.4185	11864.56/2.713	5731.43/0.9023
450	5125.1/0.2074	12166.67/2.6048	12039.41/2.6127	12049.03/2.6798	6257.1/1.0284
500	5658.93/0.2157	12335.06/2.6385	12210.78/2.7604	12220.25/2.6874	6770.68/1.0073

图 4 示出了不同放置策略下系统的能耗情况，可见 Random 算法的能耗最大，这是因为其在放置过程中开启了更多的主机；接下来是 VAS 和 MM 算法，它们按照 CPU 和内存可用资源量来选择宿主机，故也将开启更多的主机，但由于本文是模拟异构主机进行实验，因此其开启主机数略少于 Ran-

dom 算法,进而其能耗也就比 Random 算法产生的能耗少;FF 算法和本文的 DP_VMP 算法都优先考虑将虚拟机放置在已开启的主机上,故开启主机数较少,产生的能耗也较少,本文提出的 DP_VMP 算法由于考虑了主机单位能耗的放置能力以及资源的利用情况,因此产生的系统能耗最少。

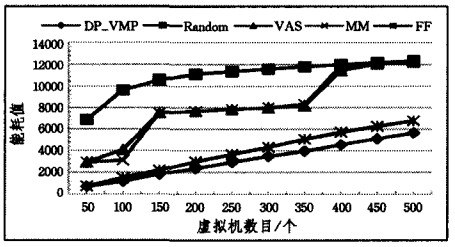

图 4　不同放置策略的能耗对比

图 5 示出了不同放置策略下系统资源损耗的情况,其中 Random 的资源损耗最大,因为它开启了更多的主机,VAS 和 MM 次之,FF 和本文的 DP_VMP 产生的损耗较少。本文的 DP_VMP 算法由于考虑了主机的资源损耗因素,因此能使整个系统的资源损耗最少。

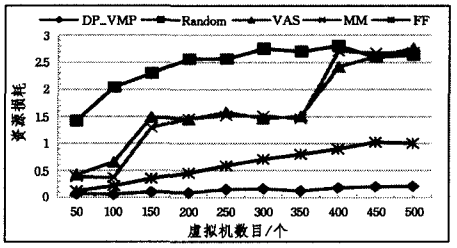

图 5　不同放置策略的资源损耗对比

图 6 示出了不同放置策略下所开启的主机个数,从图中可以看出本文所提出的 DP_VMP 策略所开启的主机数目最少;结合图 4 可发现各放置策略所开启的主机个数与其系统能耗基本成正比关系。

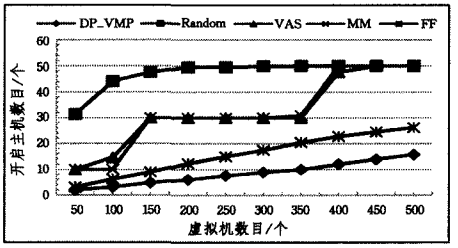

图 6　不同放置策略所开启的主机数目的对比

综上所述,若在同样的实验环境下启动相同数目的虚拟机,本文所提出的 DP_VMP 算法能获得比其他经典算法更低的系统能耗和资源损耗,同时所开启的主机数目也比其他算法的少,这充分证明了该算法在降低能耗和资源损耗方面的有效性。

结束语　本文以系统能耗和资源损耗为优化目标建立了目标模型,结合异构的云平台背景,提出了一种基于动态规划思想的多目标虚拟机放置策略 DP_VMP。该策略首先对可用物理主机进行能力评估,选出单位能耗下放置能力最强的主机,然后将虚拟机请求集合按照资源需求程度进行降序排列,最后将这些请求按照动态规划的思想进行放置处理。参照 OpenStack 云环境的仿真实验表明,该策略能够有效地降低系统的能耗和资源损耗。下一步的研究将考虑放置完成后的迁移问题,进一步优化并完善云平台系统的放置策略。

参 考 文 献

[1] CHEN K,ZHENG W M. Cloud Computing:System Instances and Current Research[J]. Journal of Software, 2009,20(5): 1337-1348. (in Chinese)
陈康,郑纬民. 云计算:系统实例与研究现状[J]. 软件学报, 2009,20(5):1337-1348.

[2] LUO L,WU W J,ZHANG F. Energy Modeling Based on Cloud Data Center[J]. Journal of Software, 2014, 25(7): 1371-1387. (in Chinese)
罗亮,吴文峻,张飞. 面向云计算数据中心的能耗建模方法[J]. 软件学报,2014,25(7):1371-1387.

[3] RICCIARDI S,CAREGLIO D,SANTOS-BOADA G,et al. Saving energy in data center infrastructures[C]// 2011 First International Conference on Data Compression, Communications and Processing(CCP). IEEE, 2011:265-270.

[4] TONG J J, HE G, FU G. Research Survey of Virtual Machine Placement Problem[J]. Computer Science, 2016, 43(s1): 249-254. (in Chinese)
童俊杰,赫罡,符刚. 虚拟机放置问题的研究综述[J]. 计算机科学,2016,43(s1):249-254.

[5] ZHANG X Y,WANG M N,DU X F. Research on the method of virtual machine deployment in cloud computing[J]. Journal of Communications,2015,36(3):241-248. (in Chinese)
张笑燕,王敏讷,杜晓峰. 云计算虚拟机部署方案的研究[J]. 通信学报,2015,36(3):241-248.

[6] AROCA J A, ANTA A F. Empirical comparison of power-efficient virtual machine assignment algorithms[J]. Computer Communications, 2016, 96: 86-98.

[7] FANG W,LIANG X,LI S,et al. VMPlanner:Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers[J]. Computer Networks, 2013,57(1):179-196.

[8] CHEN M,ZHANG H,SU Y Y,et al. Effective VM sizing in virtualized data centers[C] // 2011 IFIP/IEEE International Symposium on Integrated Network Management (IM). IEEE, 2011:594-601.

[9] PENG H, YANG G X, CAI L Z. Virtual machine deployment based on the needs of individual users[J]. Software Industry and Engineering, 2013(1): 37-41. (in Chinese)
彭红,杨根兴,蔡立志. 基于用户个性化需求的虚拟机部署机制[J]. 软件产业与工程,2013(1):37-41.

[10] ZHAO J,MA Z,LIU C,et al. Multi-objective ant colony optimization algorithm for virtual machine placement[J]. Journal of Xidian University (Natural Science), 2015, 42(3): 173-178. (in Chinese)

(下转第 75 页)

象的位置。算法的方差为：

$$\sigma_{LS}^2=\frac{28\sigma^2}{3N^2} \tag{13}$$

图 8 示出了 3 种算法的均方误差随传感器数量增加的变化关系曲线。从图 8 可见，本文提出的基于神经网络算法的性能优于最小二乘估计算法；而且当传感器节点数量增大时，其性能非常接近克拉美罗界的性能。

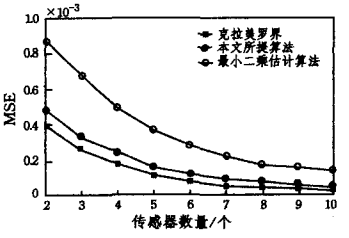

图 8　3 种算法的性能比较

结束语　本文针对 802.15.4a 无线传感器网络中的目标定位，提出了一种实现单目标和多目标的二阶段定位检测算法。该算法不仅能够实现定位误差与传感器节点和目标数量无关，而且与其他定位算法相比有更低的定位误差，从而增强了网络的鲁棒性。

鉴于传感器网络的动态复杂性，在未来的研究中将会考虑目标的移动性，因为目标的移动性会增加额外的信息，从而使得定位算法更加复杂。

参 考 文 献

[1] CATOVIC A, SAHINOGLU Z. The Cramer-Rao Bounds of Hybrid TOA/RSS and TDOA/RSS Location Estimation Schemes [J]. IEEE Communications Letters, 2004, 8(10): 626-628.

[2] GEZICI S, TIAN Z, GIANNAKIS G B, et al. Localization via Ultra-Wideband Radios[J]. IEEE Signal Processing Magazine, 2005, 22(4): 70-84.

[3] FENG C. The Research on Optimization Method for Location Accuracy of Wireless Sensor Network[D]. Nanjing: Nanjin University of Posts and Telecommunications, 2013. (in Chinese)
冯晨. 无线传感器网络定位精度优化方法研究[D]. 南京：南京邮电大学，2013.

[4] CHEN X, TANG H Y, TU S L, et al. Active distributed localization algorithm for WSN[J]. Computer Engineering and Design, 2008, 29(7): 1664-1667. (in Chinese)
陈迅，唐红雨，涂时亮，等. 无线传感器网络主动分布式节点定位算法[J]. 计算机工程与设计，2008，29(7)：1664-1667.

[5] LONG Y H, MAO H L. Single sensor detection and location method based on BP neural network[J]. Journal of Instrument and Apparatu for Analysis-monitoring, 2002(2): 20-23. (in Chinese)
龙芋宏，毛汉领. 基于 BP 神经网络的单传感器检测定位方法[J]. 仪器仪表与分析监测，2002(2)：20-23.

[6] CHANG C, SAHAI A. Object tracking in a 2D UWB sensor network[C]// Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers. New York: IEEE, 2004: 1252-1256.

[7] CHANG C, SAHAI A. Cramér-Rao-Type Bounds for Localization[J]. EURASIP Journal on Applied Signal Processing, 2006 (5): 1-13.

[8] ROVNAKOVA J, KOCUR D. UWB Radar Signal Processing for Positioning of Persons Changing Their Motion Activity[J]. Acta Polytechnica Hungarica, 2013, 10(3): 165-184.

[9] ZETIK R, SACHS J, THOMA R. UWB Localization-Active and Passive Approach[C]// Proceedings of the 21st IEEE Implementation and Measurement Technology Conference. New York: IEEE, 2004: 1005-1009.

[10] HAN Q Y. Localization Technology and Application Based on Elman Neural Networks of Wireless Sensor Networks[D]. Jinan: Shandong University of Finance and Economics, 2012. (in Chinese)
韩庆玉. 基于 Elman 神经网络的无线传感器网络定位研究与应用[D]. 济南：山东财经大学，2012.

[11] YANG K W, GUO Y B, WEI D W, et al. MFALM: An Active Localization Method for Dynamic Underwater Wireless Sensor Networks[J]. Computer Science, 2010, 37(1): 114-117. (in Chinese)
杨奎武，郭渊博，韦大伟，等. MFALM：一种水下动态传感器网络主动定位方法[J]. 计算机科学，2010，37(1)：114-117.

[12] HARDALAC F. Classification of Educational Backgrounds of Students Using Musical Intelligence and Perception with the Help of Artificial Neural Networks[J]. Expert Systems with Applications, 2009, 36(3): 6708-6713.

[13] REN F X. High-precision incremental localization algorithm for wireless sensor network[J]. Journal of Chongqing University of Posts and Telecommunications(Natural Science Edition), 2013, 25(2): 184-186. (in Chinese)
任枫轩. 高精度递增式无线传感网络的定位算法[J]. 重庆邮电大学学报(自然科学版)，2013，25(2)：184-186.

(上接第 59 页)

赵君，马中，刘驰，等. 一种多目标蚁群优化的虚拟机放置算法[J]. 西安电子科技大学学报(自然科学版)，2015，42(3)：173-178.

[11] HUANG Z N, LI H S, ZHAO J. Virtual Machine Placement Algorithm Based on Improved Genetic Algorithm[J]. Computer Science, 2015, 42(s2): 406-407. (in Chinese)
黄兆年，李海山，赵君. 基于双适应度遗传算法的虚拟机放置的研究[J]. 计算机科学，2015，42(s2)：406-407.

[12] GARG S K, VERSTEEG S, BUYYA R. A framework for ranking of cloud computing services[J]. Future Generation Computer Systems, 2013, 29(4): 1012-1023.

[13] ZHENG Q, LI R, LI X, et al. A Multi-Objective Biogeography-Based Optimization for Virtual Machine Placement[C]// 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). IEEE, 2015: 687-696.

[14] BUYYA R, BELOGLAZOV A, ABAWAJY J. Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges[C]// Proc. of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications. Las Vegas, 2010.