文章编号:1674-2974(2020)02-0116-09 DOI:10.16339/j.cnki.hdxbzkb.2020.02.016

云计算中基于多目标优化的虚拟机整合算法

胡志刚[1],肖慧[1†],李克勤[2]
(1. 中南大学 计算机学院,湖南 长沙 410083;2. 湖南大学 信息科学与工程学院,湖南 长沙 410082)

摘 要:云数据中心中存在着高能耗和高服务水平协议违约率的问题,为了解决此问题,提出了一种基于多目标优化的虚拟机整合算法.综合考虑能耗、服务质量和迁移开销等多种因素,将虚拟机整合问题构建为一个具有资源约束的多目标优化问题.使用蚁群系统算法对该多目标优化问题进行求解,进行虚拟机整合,获得近似最优的虚拟机主机映射关系.为了减少算法复杂度,利用 CPU 利用率双阈值来判断主机负载状态,根据主机负载状态分阶段进行整合并使用不同的整合策略.基于 CloudSim 平台对多目标优化的虚拟机整合算法和其他 6 种虚拟机整合算法进行仿真实验,将本文算法与现有虚拟机整合算法实验结果进行比较,结果表明本文提出的算法在能耗和服务水平协议违约方面优化显著,具有较好的综合性能.

关键词:云计算;虚拟机整合;蚁群系统算法;节能;服务质量
中图分类号:TP338.8 **文献标志码**:A

Virtual Machine Consolidation Algorithm Based on Multi-objective Optimization in Cloud Computing

HU Zhigang[1], XIAO Hui[1†], LI Keqin[2]
(1. School of Computer Science and Engineering, Central South University, Changsha 410083, China;
2. College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China)

Abstract: There exist problems of high energy consumption and high Service Level Agreement (SLA) violation rates in cloud data centers, which urgently need to be resolved. In order to solve the above problems, a Multi-objective Virtual Machine Consolidation Algorithm (MOVMC) was proposed to reduce energy consumption and SLA violation. Taking into account multiple factors including energy consumption, service quality and migration overhead, the virtual machine consolidation problem was constructed as a resource-constrained multi-objective optimization problem. Ant colony system algorithm was employed to perform virtual machine consolidation and obtain the near-optimal mapping relation between virtual machines and hosts as the solution to the multi-objective optimization problem. In order to reduce the algorithm complexity, the double thresholds of CPU utilization were leveraged to judge the host load status and a multi-stage consolidation was performed according to the host load status, in which different consolidation strategies were used. Simulation experiments were conducted on CloudSim platform for MOVMC algorithm and six other virtual machine consolidation algorithms. The experimental results show that, compared with the existing virtual machine consolidation algorithm, the proposed algorithm has significant optimization in terms of energy consumption

* **收稿日期**:2019-04-01
基金项目:国家自然科学基金资助项目(61572525),National Natural Science Foundation of China(61572525)
作者简介:胡志刚(1964—),男,山西孝义人,中南大学教授,博士生导师
† 通讯联系人,E-mail:huixiao@csu.edu.cn

and SLA violation, and an excellent comprehensive performance.

Key words: cloud computing; virtual machine consolidation; ant colony system; energy saving; quality of service

近年来，随着云计算商业模式和技术架构的越来越成熟,云用户大幅度增加,为了满足他们的需求而新建的数据中心、新置的服务器和制冷设备也随之大幅度增加，解决数据中心的高能耗问题已经成为一个大的挑战[1]. 同时,云用户对云服务的性能需求也愈加具体严格化，用户在交易前会与云服务商制定服务水平协议(Service Level Agreement,SLA)来规范化质量等级(Quality of Service,QoS)需求,以确保本次服务交易的完美达成[2]. 如果云服务商无法提供事先商定的 QoS,违背用户的期望,会给用户造成不可预估的损失. 因此，在减少数据中心能耗的同时，提供用户所期望的 QoS 是云计算发展迫切需要解决的问题.

虚拟机整合[3]可以根据变化的资源需求周期性地调整当前的虚拟机主机间映射关系，在主机间迁移虚拟机以充分并均衡地利用计算资源. 虚拟机整合技术主要包括启发式贪心算法[4-7]、线性/约束规划技术[8-11]和元启发式算法[12-15]. 贪心算法因其时间复杂度低、实现简单等优点被广泛应用来进行虚拟机动态整合. 贪心算法虽然计算开销低,但却容易陷入局部最优而错过最优解. 线性/约束规划技术可以获得最优解,但受问题规模和复杂性的限制,无法很好地扩展到大型数据中心. 近年来研究人员提出了许多基于生物启发计算的元启发整合算法，例如蚁群算法、基因算法、人工蜂群算法,可以有效帮助解决大规模问题并避免局部最优解. 蚁群系统算法(Ant Colony System,ACS)[16-17],作为蚁群算法的一种,通过在解空间中进行基于概率式的搜索，可以在多项式时间复杂度里找到近似最优解.

现有的虚拟机整合研究大多只关注了云数据中心的能耗问题. 然而，为了实现云系统交付的 QoS,还应该同时考虑 SLA 违约问题. 虚拟机整合可以通过将虚拟机整合到尽可能少的主机上来降低能耗,然而过分整合可能会降低系统性能并导致 SLA 违约[18]. 因此,最优虚拟机整合方法应在能耗和 QoS 之间取得平衡.

本文将虚拟机整合问题构建为一个多目标组合优化问题,优化目标包括降低能耗、保证 QoS 要求和减少迁移次数，提出了一种基于多目标优化的虚拟机整合算法(Multi-objective Virtual Machine Consolidation,MOVMC). 首先使用 CPU 利用率双阈值 [4]来判断主机负载状态,确定整合时机;然后基于 ACS 假设虚拟机和主机之间的映射关系是食物源，使用人工蚁群同时选择待迁移虚拟机和目标主机，寻找虚拟机和主机之间的最佳映射关系. 通过在 CloudSim 平台上使用真实工作负载来评估所提出的方法. 实验结果表明,该方法在减少能耗、SLA 违约和虚拟机迁移方面具有明显的优势.

1　基于多目标优化的虚拟机整合模型

1.1　数据中心模型

在数据中心中有许多不同配置的主机，虚拟机根据各自的资源需求被分别部署到合适的主机上运行. 将数据中心里的主机集群记为:

$$H=\{h_1,h_2,\cdots,h_j,\cdots,h_m\}$$

其中 m 为主机的数量. 数据中心中运行的虚拟机集合记为:

$$V=\{v_1,v_2,\cdots,v_i,\cdots,v_n\}$$

其中 n 为虚拟机的数量. 变量 x_{ij} 表示虚拟机 v_i 是否分配给主机 h_j，若已分配则值为 1，否则为 0. 矩阵 $\boldsymbol{X}=[x_{i,j}]_{n\times m}$ 表示数据中心里虚拟机和主机间的映射关系. 部署了虚拟机 v_i 的主机记为 $h(v_i)$,主机 h_j 上运行的虚拟机集合记为 $V\ (h_j)$. 虚拟机 v_i 申请的 CPU 资源量记为 VR_j，主机 h_j 总 CPU 资源需求量 PR_j 可表达为:

$$\mathrm{PR}_j=\sum_{i=1}^{N}(x_{ij}\cdot\mathrm{VR}_j) \tag{1}$$

主机 h_j 的 CPU 资源总量记为 CR_j,则主机 h_j 的 CPU 资源利用率 U_j 可通过式(2)计算得到:

$$U_j=\frac{\mathrm{PR}_j}{\mathrm{VR}_j} \tag{2}$$

1.2　能耗模型

数据中心中主机的能耗与工作负载程度有关,取决于 CPU、内存、磁盘和网卡等资源的利用率. 大多数研究表明，主机的能耗主要来自 CPU，和 CPU

利用率呈高度正相关关系[19-20]. 基于此,数据中心总能耗可以表示为:

$$\mathrm{EC}=\sum_{j=1}^{m}\int_{t}[\mathrm{EC}_j(U_j)]\mathrm{d}t \tag{3}$$

式中:$\mathrm{EC}_j(U_j)$表示主机 h_j 随 CPU 利用率 U_j 变化而产生的能耗.

1.3 多目标组合优化

综合考虑影响虚拟机整合的多个因素,本文定义了以下虚拟机整合优化目标.

为了降低能耗和节约能源,应该尽量减少数据中心总能耗 EC,即 min(EC).

同一主机上的虚拟机间存在资源竞争. 以往研究表明[21],当资源竞争越激烈,尤其当资源利用率接近或超过一定阈值时,主机的性能越差,QoS 水平越容易下降. 显然,主机的 QoS 水平与资源利用率有很大的关系. 因此为了更方便有效地优化 QoS,通过限制主机 CPU 资源利用率低于过载阈值 $\mathrm{Thr_o}$,以保证主机 QoS,如式(4)所示:

$$U_j<\mathrm{Thr_o},\ \forall j\in\{1,2,\cdots,m\} \tag{4}$$

另外,虚拟机整合触发的虚拟机迁移会消耗能源和计算资源,给网络传输造成负担,也会影响数据中心里其他运行虚拟机的性能[22],因此需尽可能减少虚拟机迁移次数 MG,即 min(MG).

为了优化以上目标,通过线性加权的方式将其整合到一起,简化表达为一个有着资源限制的最小组合优化问题,如式(5)所示:

$$\begin{aligned}&\min(F)=\min(\mathrm{EC}+\omega_{\mathrm{mg}}\cdot\mathrm{MG})\\&\text{s.t.}U_j<\mathrm{Thr_o},\ \forall j\in\{1,2,\cdots,m\}\end{aligned} \tag{5}$$

式中:ω_{mg} 代表迁移次数 MG 相对于总能耗 EC 的相对权重.

2 MOVMC 整合算法

虚拟机动态整合支持虚拟机在运行的主机之间动态迁移,来适应数据中心里不断变动的负载. 对于任何一种虚拟机整合算法,有 3 个关键问题必须加以解决[23]. 第一,虚拟机整合在什么条件下会被触发(选择整合时机);第二,选择哪些虚拟机迁移能达到最好的能耗和 QoS(选择迁移虚拟机);第三,将选择迁移的虚拟机重新部署到哪些主机上最好(选择目标主机).

本文提出的 MOVMC 整合算法利用 CPU 利用率双阈值和 ACS 来解决上述问题. 基于静态双阈值策略,MOVMC 算法通过设置主机 CPU 利用率上限阈值 $\mathrm{Thr_o}$ 和下限阈值 $\mathrm{Thr_u}$($0\leqslant\mathrm{Thr_u}<\mathrm{Thr_o}\leqslant1$)来确定整合时机. 利用设置的静态阈值识别主机的负载状态,将所有主机划分为 3 个集合,过载主机集合 H_{o}、正常负载主机集合 H_{n}、低载主机集合 H_{u}. 对主机 H_j 而言,当 $\mathrm{Thr_o}\leqslant U_j<1$ 时,主机过载;当 $\mathrm{Thr_u}\leqslant U_j<\mathrm{Thr_o}$ 时,主机为正常负载状态;当 $0<U_j<\mathrm{Thr_u}$ 时,主机低载. 当主机过载或低载时需要进行虚拟机整合. 根据 Beloglazov[4]等的研究并通过实验验证得到:当设置上限阈值 $\mathrm{Thr_o}$ 为 80%,下限阈值 $\mathrm{Thr_u}$ 为 40%时,算法可以获得优秀的性能.

定义映射关系元组集合 $T=\{(v_{\mathrm{m}},h_{\mathrm{d}})\}$,$T$ 中每个元组包括两个元素:选择迁移的虚拟机 v_{m} 和目标主机 h_{d}. MOVMC 算法将 T 中的元组看作食物源,使用 ACS 搜索 T 中的元组,来更新虚拟机主机映射关系.

虚拟机整合算法的执行时间主要取决于 T 中元组的数量. 为了减少算法的时间复杂度,进行多阶段整合,限制 T 中元组的数量. 一方面,先对过载主机进行虚拟机整合,再对低载主机进行整合. 另一方面,在选择目标主机时,为了尽量减少低载主机数量以降低能耗,首先在正常负载主机集合中选择,若寻找目标主机失败,则在低载主机集合中选择. 只有当无法将虚拟机迁移到正在运行的主机上时,才会激活处于睡眠模式的主机. 这样可以缩小每次搜索的解空间,提高蚂蚁搜索解的效率,在不影响解质量的情况下,能大大减少算法的计算时间.

在虚拟机整合过程中,MOVMC 算法创建信息素矩阵 $[\tau_{i,j}]_{n\times m}$ 来保存过去搜索的经验,$\tau_{i,j}$ 表示选择映射关系元组(v_i,h_j)的有利度. 使用局部信息素更新和全局信息素更新两种规则来更新元组信息素. 定义启发式因子来基于伪随机比例规则指导蚂蚁选择最优元组,获得最小能耗和最少迁移次数. 在下文中,将给出以上概念的详细定义.

2.1 信息素踪迹

蚂蚁在寻找食物源的途中会沉积信息素,其他蚂蚁闻到信息素后,会倾向于选择信息素浓度较高的途径. 在初始化信息素矩阵$[\tau_{i,j}]_{n\times m}$时,采用 Dorigo 等人[17]使用的解质量评估方法来计算信息素的初始值,定义为:

$$\tau_0=\frac{1}{\mathrm{EC_0}\cdot\mathrm{MG}} \tag{6}$$

式中:$\mathrm{EC_0}$ 是初始状态下所有主机的功耗;MG 是近似最优的迁移次数,其值可以通过最近邻启发式算法[17]估计得到.

在每次选择新元组(v_i,h_j)后,蚂蚁会使用以下局部信息素更新规则来更新遍历元组的信息素水平:

$$\tau_{ij}=(1-\rho_l)\cdot\tau_{ij}+\rho_l\cdot\tau_0 \tag{7}$$

式中:$\rho_l\in[0,1]$为局部信息素消散参数.

在所有蚂蚁完成解的构建后,根据目标函数评估当前所有解的质量. 为了保存蚂蚁以往搜索过程中获得的经验,针对全局最优解应用以下全局信息素更新规则:

$$\begin{cases}\tau_{ij}=(1-\rho_g)\cdot\tau_{ij}+\rho_g\cdot\Delta\tau\\ \Delta\tau=\begin{cases}\dfrac{1}{F(X^+)},\text{if } x_{ij}=1 \text{ in } X^+\\ 0,\text{其他}\end{cases}\end{cases} \tag{8}$$

式中:$\Delta\tau$ 为增加的额外信息素量;$\rho_g\in[0,1]$是全局信息素消散参数;X^+是算法目前找到的最优解.

2.2 启发式因子

在 ACS 中,启发式因子被用来与信息素相结合,共同指导蚂蚁构建解方案. 启发式因子用 $\eta_{i,j}$ 表示,代表选择元组(v_i,h_j)的期望值.

在考虑虚拟机选择和目标主机选择的基础上,定义一种选择元组(v_i,h_j)的混合启发式因子 $\eta_{i,j}$ 为:

$$\eta_{i,j}=\eta^v(h(v_i),-v_i)\cdot\eta^h(h_j,+v_i) \tag{9}$$

式中:$\eta^v(h(v_i),-v_i)$是从 v_i 的源主机 $h(v_i)$选择迁移 v_i 的启发式因子,而 $\eta^h(h_j,+v_i)$是将 v_i 重新部署到目标主机 h_j 的启发式因子.

2.2.1 选择虚拟机

当主机过载时,必须从主机迁出某些虚拟机以释放部分资源以防止 SLA 违约. 迁移 CPU 利用率高的虚拟机可以尽量缓解主机的过载危机,减少虚拟机迁移次数. 对过载主机 $h_j\in H_o$,定义选择虚拟机 v_i 迁移的启发式因子如下:

$$\eta^v(h_j,-v_i)=\frac{\mathrm{VR}_j}{\mathrm{PR}_j} \tag{10}$$

当主机处于低载状态时,应该迁出该主机上的所有虚拟机,并将其切换到休眠状态,以避免空闲功耗. 因此,为了尽可能减少无效迁移和低载主机数量,应优先选择迁移虚拟机,使低载主机迁出后资源利用率尽量低. 因此定义选择低载主机 $h_j\in H_u$ 中虚拟机 v_i 迁移的启发式因子为:

$$\eta^v(h_j,-v_i)=1-U_j(-v_i) \tag{11}$$

式中:$U_j(-v_i)$代表主机 h_j 迁出虚拟机 v_i 后的 CPU 利用率.

2.2.2 选择目标主机

为了避免主机 CPU 利用率过高导致的 QoS 降级,在正常负载主机集合中选择目标主机时,选择迁入虚拟机后资源利用率低的主机更有利. 为虚拟机 v_i 选择目标主机 $h_j\in H_n$ 的启发式因子定义如下:

$$\eta^h(h_j,+v_i)=\begin{cases}1-U_j(+v_i),\text{if } U_j(+v_i)<\mathrm{Thr}_o\\ 0,\text{其他}\end{cases} \tag{12}$$

式中:$U_j(+v_i)$是主机 h_j 迁入虚拟机 v_i 后的 CPU 利用率. 另外在启发式因子中设置了 CPU 资源利用率约束,以防止导致目标主机过载的迁移.

低负载主机的资源利用率较低,可以保证 QoS 却十分浪费能源. 因此,在低负载主机集合 H_u 中选择目标主机 h_j 时,优先选择迁入虚拟机之后资源利用率更高的主机,以最小化低载主机数量,对应的启发式因子定义如下:

$$\eta^h(h_j,+v_i)=\begin{cases}U_j(+v_i),\text{if } U_j(+v_i)<\mathrm{Thr}_o\\ 0,\text{其他}\end{cases} \tag{13}$$

2.3 伪随机比例规则

基于启发式因子和信息素信息,蚂蚁使用伪随机比例规则来选择下一个元组(v_m,h_d)进行遍历,如式(14)所示:

$$(v_m,h_d)=\begin{cases}\arg\max_{(v_i,h_j)\in\Omega_k}\{\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}\},\text{if } q\leqslant q_0\\ (v_s,h_g),\text{其他}\end{cases} \tag{14}$$

式中:α 和 β 分别为控制信息素和启发式因子的重要性参数;q 为一个均匀分布在[0,1]的随机数;$q_0\in[0,1]$是一个固定的参数,决定累积经验与随机选择的相对重要性;(v_s,h_g)是根据式(15)中的概率分布选择的随机元组变量.

$$p_{md}^{k}=\begin{cases}\dfrac{\tau_{md}^{\alpha}\cdot\eta_{md}^{\beta}}{\sum_{(v_i,h_j)\in\Omega_k}(\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta})},\text{if } (v_j,h_j)\in\Omega_k\\ 0,\text{其他}\end{cases} \tag{15}$$

式中:p_{md}^{k}表示蚂蚁 ant_k 下一步选择遍历元组(v_m,h_d)的概率;Ω_k 为当前允许被蚂蚁 ant_k 遍历的元组集合.

结合上文中提出的启发式因子计算准则和伪随机比例规则,本文提出一种选择元组算法(Next Tuple Selection,NTS),该算法基于源主机和目标主机负载状态标识 heuP 选择下一个映射关系元组(v_m,h_d)进行遍历,算法伪代码如算法 1 所示. 本文提出的虚拟机整合算法 MOVMC 的伪代码如算法 2 所示. MOVMC 算法通过限制蚁群算法搜索的解集合大小有效降低了算法复杂度. 从算法 2 的伪代码可以看出,MOVMC 在最坏情况下的算法复杂度是 O(nI×|$V(H_o)$|×|T|+nI×|$V(H_u)$|×|T|),其中 nI 是迭代的次数,|$V(H_o)$|和|$V(H_u)$|分别表示过载主机和低载主机中虚拟机的总数量. 如算法 2 中第 3 行所示,最外层循环迭代 nI 次. 因为蚂蚁可以并行搜索各自的解,所以算法 2 中第 4 行所示的第 2 层循环不会对

算法的时间复杂度造成影响. 如算法 2 中第 5 行和 18 行所示,在最坏情况下,第 3 层循环需要分别迭代 $|V(H_o)|$ 和 $|V(H_u)|$ 次. 在每次迭代时, 如算法 1 所示, 需要循环计算 $|T|$ 个元组对应的概率以选择下一个元组进行遍历.

算法 1 选择元组进行遍历(NTS)

输入:T,heuP
输出:(v_m,h_d)
1. switch(heuP)
2. //选择虚拟机从过载主机迁移到正常负载主机
3. case "ON"
4. 根据式(10)(12)(9)计算 $\eta=\{\eta_{md}|\forall(v_m,h_d)\in T\}$
5. break
6. //选择虚拟机从过载主机迁移到低负载主机
7. case "OU"
8. 根据式(10)(13)(9)计算 $\eta=\{\eta_{md}|\forall(v_m,h_d)\in T\}$
9. break
10. //选择虚拟机从低载主机迁移到正常负载主机
11. case "UN"
12. 根据式(11)(12)(9)计算 $\eta=\{\eta_{md}|\forall(v_m,h_d)\in T\}$
13. break
14. //选择虚拟机从低载主机迁移到低负载主机
15. case "UU"
16. 根据式(11)(13)(9)计算 $\eta=\{\eta_{md}|\forall(v_m,h_d)\in T\}$
17. break
18. 根据式(15)计算 $P_{vmc}=\{p_{md}|\forall(v_m,h_d)\in T\}$
19. 根据式(14)选择下一个遍历的元组 $(v_m,h_d)\in T$

算法 2 MOVMC

输入:X
输出:X^+
1. $S\leftarrow\phi$,$X^+\leftarrow\phi$,$X^k\leftarrow\phi$,$t\leftarrow\phi$
2. 根据式(6)将信息素矩阵初始化为 τ_0
3. for $i\in[1,nI]$ do
4. for $k\in[1,nA]$ do
5. for $v_o\in V(H_o)$ do //过载主机虚拟机整合
6. $T_{on}\leftarrow\{(v_m,h_d)|h(v_m)\in H_o\wedge H_d\in H_n\}$
7. $t\leftarrow$ NTS(T_{on},"ON")
8. if t 为空 then
9. $T_{ou}\leftarrow\{(v_m,h_d)|h(v_m)\in H_o\wedge H_d\in H_u\}$
10. $t\leftarrow$ NTS(T_{ou},"OU")
11. if t 为空 then
12. break
13. end if
14. end if
15. 更新虚拟机主机映射关系
16. 根据式(7)对 t 应用局部信息素更新规则
17. end for
18. for $v_u\in V(H_u)$ do//低载主机虚拟机整合
19. $T_{un}\leftarrow\{(v_m,h_d)|h(v_m)\in H_u\wedge H_d\in H_n\}$
20. $t\leftarrow$ NTS(T_{un},"UN")
21. if t 为空 then
22. $T_{uu}\leftarrow\{(v_m,h_d)|h(v_m)\in H_u\wedge H_d\in H_u\}$
23. $t\leftarrow$ NTS(T_{uu},"UU")
24. if t 为空 then
25. break
26. end if
27. end if
28. 更新虚拟机主机映射关系
29. 根据式(7)对 t 应用局部信息素更新规则
30. end for
31. $S\leftarrow S\cup\{X^k\}$
32. end for
33. $X^+\leftarrow\arg\max_{X^k\in S}\{F(X^k)\}$
34. 根据式(8)对 X^+应用全局信息素更新规则
35. end for

3 实验与算法评估

3.1 实验环境和配置

为了评估算法性能,采用 CloudSim 模拟云计算环境[24]进行仿真实验. CloudSim 是一个离散型事件模拟器,支持虚拟环境建模和虚拟资源管理. 实验中模拟的云数据中心包含 800 台物理主机, 主机有两种配置,一半为 HP ProLiant G4,另一半为 HP ProLiant G5. 主机配置详情见表 1. 此外, 本次实验使用了 4 种 Amazon EC2 虚拟机[25],虚拟机实例详情如表 2 所示. 实验中使用的工作负载来自 CoMon 项目,该项目主要负责监控 PlanetLab[26]中基础设施的运行情况. 实验采用的数据从遍布全球 500 多个地区超过 1 000 个虚拟机上采集获得.

表 1 主机配置

Tab.1 Host configuration

主机	CPU 类型	频率/GHz	内核个数	RAM/GB
HP ProLiant G4	Intel Xeon 3040	1.86	2	4
HP ProLiant G5	Intel Xeon 3075	2.66	2	4

表 2 虚拟机类型

Tab.2 VM types

虚拟机类型	CPU 频率/MIPS	RAM/GB
超强 CPU 基本型	2 500	0.85
超大实例	2 000	3.75
小型实例	1 000	1.70
微型实例	500	0.61

3.2 评估指标

3.2.1 能耗指标

随着数据中心内主机硬件设施不断升级，例如多核处理器,大容量内存和硬盘,传统的 CPU 利用率–能耗模型已经无法再准确地描述能耗. 为了得到更准确的能耗值，使用标准性能评估公司(Standard Performance Evaluation Corporation)提供的真实数据. 表 3 显示了不同负载级别下 HP ProLiant G4 和 HP ProLiant G5 主机的能耗.

表 3　主机在不同 CPU 利用率级别下的能耗

Tab.3　Host energy consumption at different CPU utilization levels

CPU 利用率/%	能耗/W	
	HP ProLiant G4	HP ProLiant G5
0	86	93.7
10	89.4	97
20	92.6	101
30	96	105
40	99.5	110
50	102	116
60	106	121
70	108	125
80	112	129
90	114	133
100	117	135

3.2.2 SLA 违约指标

在数据中心里，保证用户所需 QoS 十分重要，SLA 规定了用户的 QoS 需求. 为了更好地优化 QoS,Beloglazov 等[5]提出了若干衡量 SLA 违约的度量标准.

SLAVO 定义为主机资源利用率达到 100%的时间所占的比例,以衡量主机过载导致的 SLA 违约,如式(16)所示:

$$\mathrm{SLAVO}=\frac{1}{m}\sum_{j=1}^{m}\frac{T_j^{\mathrm{o}}}{T_j^{\mathrm{a}}} \tag{16}$$

式中:m 表示主机的数量;T_j^{o} 表示主机 h_j 资源利用率达到 100%的总时间;T_j^{a} 为主机 h_j 运行的总时间.

虚拟机迁移会导致总体性能下降. SLAVM 代表因虚拟机迁移而造成的 SLA 违约,定义为:

$$\mathrm{SLAVM}=\frac{1}{n}\sum_{i=1}^{n}\frac{C_i^{\mathrm{d}}}{C_i^{\mathrm{r}}} \tag{17}$$

式中:n 表示虚拟机的数量;C_i^{d} 代表由虚拟机 v_i 迁移引起的性能降级的估计值;C_i^{r} 为虚拟机 v_i 生命周期内的总 CPU 资源需求量. 根据以往研究[5],将因虚拟机迁移而导致的开销 C_i^{d} 设定为该虚拟机 CPU 利用率的 10%.

SLAV 用于评估云数据中心的整体 QoS 水平,能综合反映出由主机超载和虚拟机迁移导致的总性能下降,定义如下:

$$\mathrm{SLAV}=\mathrm{SLAVO}\times\mathrm{SLAVM} \tag{18}$$

SLAV 变量值越小，说明 SLA 违约越少,QoS 水平越高.

3.2.3 能耗–性能指标

虚拟机整合应该平衡地对能耗和 SLA 违约进行优化. 通过结合能耗指标 EC 和 SLA 违约指标 SLAV 获得综合的能耗–性能指标 ESV,如式(19)所示:

$$\mathrm{ESV}=\mathrm{EC}\times\mathrm{SLAV} \tag{19}$$

式中:EC 表示数据中心总能耗. EC 值越低，表明数据中心能源效率越高.

3.2.4 虚拟机迁移次数

虚拟机的动态迁移会产生一定的开销，例如占用计算资源和消耗能源. 此外,由于虚拟机迁移导致的服务暂停可能会影响 QoS. 因此,减少虚拟机迁移数量可以节约资源和能耗并提高 QoS，有助于虚拟机整合获得理想的效果.

3.3 参数选择

在 MOVMC 算法中,α 和 β 值分别控制蚁群算法中信息素和启发式因子的重要性，对蚁群算法的性能有很大的影响. α 反映了蚂蚁在以往搜索过程中积累的信息素在指导蚁群寻找新解时的重要程度. α 值越大,蚂蚁越有可能选择之前走过的路径,寻找随机解的可能性随之降低. β 则反映了启发式因子在指导蚁群寻找新解时的重要程度. β 值越大，启发式因子的重要程度越大. α 和 β 的取值越大,计算量也越大,计算时间就越长. 考虑到数据中心内庞大的数据量，为了得到最优的算法性能，选择 α 和 β 值在[0.5,3.5]内以 0.5 为单位递增,讨论 α 和 β 的组合取值对算法目标函数值的影响，实验结果如图 1 所示. 算法中使用的其他参数值如表 4 所示. 当 α 和 β 取值过大时,算法容易过早收敛,易陷入局部最优. 当 α 和 β 取值过小时，基于经验值和先验性因素的指导作用减弱,致使算法很难找到最优解. 从图 1 中可以看出,当 α 和 β 取值分别为 1 和 1.5 时,对应的目标函数值最小,系统可以取得最好的性能. 故在后面的对比实验中，将 α 和 β 分别取值为 1 和 1.5 作为

系统参数值.

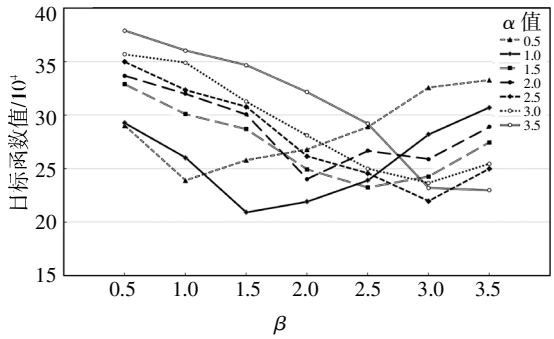


图 1 不同 α 和 β 取值对比

Fig.1 Comparison of different α and β

表 4 MOVMC 算法参数

Tab.4 MOVMC parameters

ω_{mg}	q_0	ρ_l	ρ_g	nA	nI
9	0.9	0.1	0.1	10	10

3.4 结果分析

为了评估 MOVMC 算法的性能，本研究将 MOVMC 算法与 6 种虚拟机动态分配算法 ST[4]、DT[4]、MAD[5]、IQR[5]、LR[5]和 EVMCACS[15]进行比较. 图 2 为在实际工作负载下 MOVMC、ST、DT、MAD、IQR、LR、EVMCACS 的能耗值. 算法名称后面的数值为该算法的当前参数值. 与 ST、DT、MAD、IQR、LR、EVMCACS 相比，MOVMC 消耗的能源最少. 由于 MOVMC 算法优先将虚拟机迁移到正常负载的主机，因此许多低载主机都可以切换到睡眠模式，从而减少了数据中心中活跃主机的数量并节省大量能源.

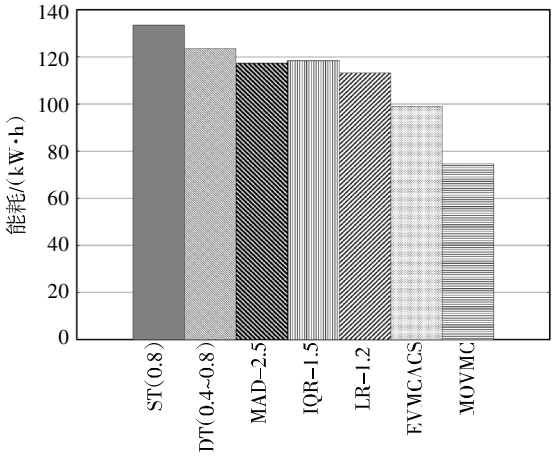
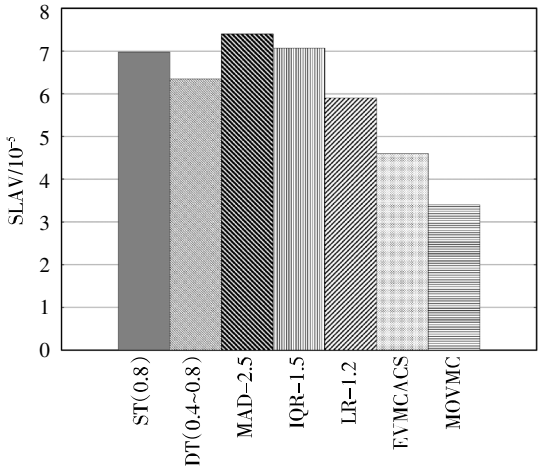


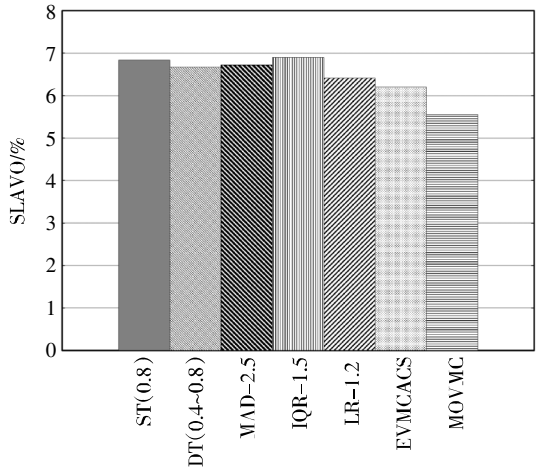
图 2 能耗值对比

Fig.2 Comparison of energy consumption

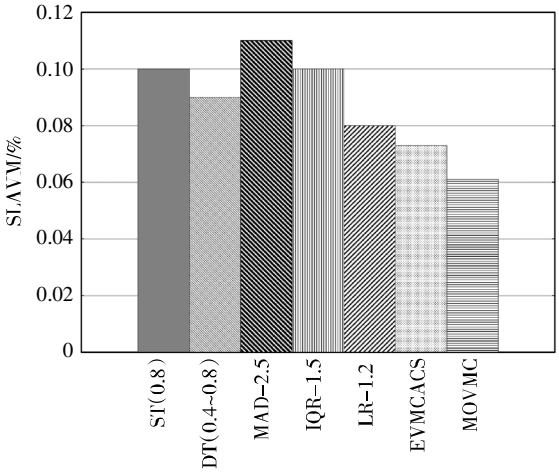
图 3 是算法针对 SLA 违约指标进行的比较. 图 3(a) 是对 SLAV 的比较，可以看出，与其他算法相比，MOVMC 算法在实际工作负载下具有最低的 SLA 违约率. MOVMC 算法从过载主机中迁出虚拟机，通过保证主机的 CPU 利用率低于过载阈值来防止 SLA 违约. 另外，启发式准则确保目标主机在迁入虚拟机后不会超出过载阈值. 因此，MOVMC 算法的 SLAV 性能优于其他算法.



(a) SLAV 值对比



(b) SLAVO 值对比



(c) SLAVM 值对比

图 3 SLA 性能指标值对比

Fig.3 Comparison of SLA performance indexes

SLAV 是综合 SLAVO 和 SLAVM 得到的指标,因此接下来分别对算法的 SLAVO 和 SLAVM 进行分析. 图 3(b)为 SLAVO 值的对比,可以清楚地看出 MOVMC 算法的 SLAVO 值低于其他算法. 这主要是因为 MOVMC 算法将主机的资源利用率限制在过载阈值以下,有效地降低了主机资源过载的风险,保证了运行主机的 QoS. 此外, 在重新部署虚拟机时, MOVMC 算法优先选择正常负载主机中资源利用率更低的主机, 从另一个角度保证了主机运行时的 QoS. 图 3(c)为不同算法的 SLAVM 性能比较. 从图中可以看出,MOVMC 算法在 SLAVM 方面的性能最优. 根据图 4,MOVMC 算法有效地减少了虚拟机迁移次数, 从而减少了迁移对服务质量的影响. 因此,与其他算法相比,MOVMC 算法具有较低的 SLAVM.

由图 4 可知,MOVMC 算法提高了迁移效率并获得了最少的迁移次数. MOVMC 算法有效地保证了运行主机良好的 QoS,降低了主机过载风险,从而减少了主机过载触发的虚拟机迁移数量. 另一方面,MOVMC 算法中定义的目标函数趋于最少化虚拟机迁移的次数.

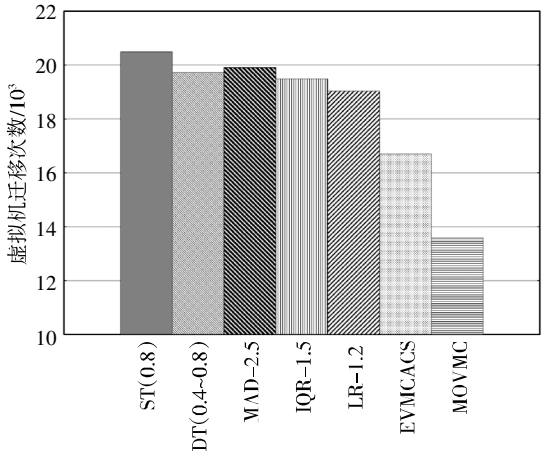


图 4　虚拟机迁移次数对比

Fig.4　Comparison of the number of VM migrations

图 5 展示了利用 ESV 指标进行评估得到的算法综合性能,可以看出 MOVMC 算法综合性能最好. 根据仿真结果,MOVMC 算法的 ESV 指标值仅是 DT 算法的 33%,ST 算法的 ESV 指标最差. 主要原因是 MOVMC 算法通过精确识别主机负载状态,有效降低了主机过载风险和虚拟机迁移次数, 减少了低载主机的数量.

以上实验结果表明,MOVMC 算法能做出更合理的虚拟机迁移决策, 在降低数据中心能耗和保证 QoS 方面有较好的性能.

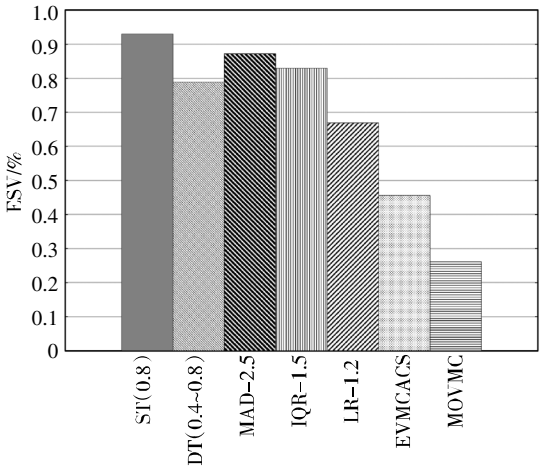


图 5　ESV 值对比

Fig.5　Comparison of ESV

4　结　论

本文提出了一种基于多目标组合优化的虚拟机整合算法, 通过将虚拟机整合到合适的主机中来解决数据中心中高能耗和 QoS 降级的问题. 虚拟机整合问题被构建为一个多目标优化问题, 基于双阈值决定触发虚拟机整合的条件. 将虚拟机与主机之间的映射关系比作食物源, 基于 ACS 通过多阶段整合来优化映射关系. 通过人工蚂蚁的分布式搜索和协作,获得虚拟机与主机之间的全局最优映射关系. 使用实际工作负载对所提出方法的性能进行评估,仿真结果表明,与其他方法相比,该方法能有效降低数据中心的能耗,并保证高水平 QoS.

在未来的工作中, 进一步研究在整合时间决策过程中,针对不断变化的工作负载采用自适应阈值,做出更合理的迁移决策. 进行更多的仿真实验来评估所提出的方法在实际工作负载中的性能.

参考文献

[1] 蔡立军,何庭钦,孟涛,等. 基于层次拓扑树的虚拟机节能分配算法[J].湖南大学学报(自然科学版),2017,44(2):137—148.
CAI L J,HE T Q,MENG T,*et al*. A network-aware two-phase virtual machine allocation algorithm [J]. Journal of Hunan University (Natural Sciences),2017,44 (2):137—148. (In Chinese)

[2] ARDAGNA D,CASALE G,CIAVOTTA M,*et al*. Quality-of-service in cloud computing:modeling techniques and their applications[J]. Journal of Internet Services & Applications,2014,5(1):11—17.

[3] FILHO M C S,MONTEIRO C C,INÁCIO P R M,*et al*. Approaches for optimizing virtual machine placement and migration in cloud

environments:A survey [J]. Journal of Parallel & Distributed Computing,2017,111:222—250.

[4] BELOGLAZOV A,ABAWAJY J,BUYYA R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing [J]. Future Generation Computer Systems, 2012,28(5):755—768.

[5] BELOGLAZOV A,BUYYA R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers [J]. Concurrency and Computation:Practice and Experience,2012,24(13):1397—1420.

[6] ZHOU Z,ABAWAJY J,CHOWDHURY M,*et al*. Minimizing SLA violation and power consumption in cloud data centers using adaptive energy-aware algorithms [J]. Future Generation Computer Systems,2018,86:836—850.

[7] LI M F,BI J P,LI Z C. Improving consolidation of virtual machine based on virtual switching overhead estimation [J]. Journal of Network and Computer Applications,2016,59:158—167.

[8] CHEN X,TANG J R,ZHANG Y. Towards a virtual machine migration algorithm based on multi-objective optimization [J]. International Journal of Mobile Computing & Multimedia Communications,2017,8(3):79—89.

[9] CHEN L H,SHEN H Y,PLATT S. Cache contention aware virtual machine placement and migration in cloud datacenters [C]// 2016 IEEE 24th International Conference on Network Protocols (ICNP). Singapore: IEEE,2016:1—10.

[10] JOO K N,KIM S,KANG D K,*et al*. A VM vector management scheme for QoS constraint task scheduling in cloud environment[C] // International Conference on Cloud Computing. Korea:Springer International Publishing,2015:39—49.

[11] HUANG Z,TSANG D H K. M-convex VM Consolidation:Towards a better VM workload consolidation [J]. IEEE Transactions on Cloud Computing,2016,4(4):415—428.

[12] LI Z,YAN C,YU L,*et al*. Energy-aware and multi-resource overload probability constraint-based virtual machine dynamic consolidation method [J]. Future Generation Computer Systems, 2018,80:139—156.

[13] MOSA A,PATON N W. Optimizing virtual machine placement for energy and SLA in clouds using utility functions [J]. Journal of Cloud Computing,2016,5(1):1—17.

[14] LI H J,ZHU G F,CUI C Y,*et al*. Energy-efficient migration and consolidation algorithm of virtual machines in data centers for cloud computing[J]. Computing,2016,98(3):303—317.

[15] ARYANIA A,AGHDASI H S,KHANLI L M. Energy-aware virtual machine consolidation algorithm based on ant colony system [J]. Journal of Grid Computing,2018,16(3):477—491.

[16] DORIGO M,CARO G D,GAMBARDELLA L M. Ant algorithms for discrete optimization[J]. Artificial Life,1999,5(2):137—172.

[17] DORIGO M,GAMBARDELLA L M. Ant colony system:A cooperative learning approach to the traveling salesman problem[J]. IEEE Transactions on Evolutionary Computation,1997,1(1):53—56.

[18] ZHAO H,WANG J,LIU F,*et al*. Power-aware and performance-guaranteed virtual machine placement in the cloud [J]. IEEE Transactions on Parallel & Distributed Systems,2018,29 (6): 1385—1400.

[19] 吴小东，韩建军. 云数据中心基于阈值的虚拟机迁移节能调度算法 [J]. 华中科技大学学报（自然科学版),2018,46(9):30—34.
WU X D,HAN J J. Threshold-based energy-efficient VM scheduling in cloud datacenters[J]. Journal of Huazhong University of Science and Technology (Natural Sciences),2018,46(9):30—34. (In Chinese)

[20] ZHENG Q H,LI R,LI X Q,*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. Shenzhen:IEEE,2015:687—696.

[21] CHEN Q,CHEN J X,ZHENG B Y,*et al*. Utilization-based VM consolidation scheme for power efficiency in cloud data centers[C]// IEEE International Conference on Communication Workshop. London:IEEE,2015:1928—1933.

[22] LI Z H,YAN C Y,YU X R,*et al*. Bayesian network-based Virtual Machines consolidation method [J]. Future Generation Computer Systems,2018,16(3):477—491.

[23] ZHANG F,LIU G,FU X,*et al*. A survey on virtual machine migration:challenges,techniques and open issues [J]. IEEE Communications Surveys & Tutorials,2018,20(2):1206—1243.

[24] CALHEIROS R N,RANJAN R,BELOGLAZOV A,*et al*. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms[J]. Software:Practice and Experience,2011,41(1):23—50.

[25] DURAO F,CARVALHO J F S,FONSEKA A,*et al*. A systematic review on cloud computing [J]. The Journal of Supercomputing, 2014,68(3):1321—1346.

[26] PARK K S,PAI V S. COMON:A mostly-scalable monitoring system for planetlab [J]. ACM Sigops Operating Systems Review,2006,40 (1):65—74.