基于渗透式人工蜂群与蚁群优化混合的负载平衡算法 *

侯 翔，杨成福，刘笃晋

(四川文理学院, 智能制造学院, 四川 达州 635000)

摘 要：针对当前云计算负载平衡调度过程中出现的虚拟机迁移效率低和能耗高问题，提出了一种基于渗透式人工蜂群与蚁群混合优化负载平衡算法，该算法将化学渗透行为与生物启发的负载平衡算法相结合，在充分利用人工蜂群和蚁群两种优化算法优点的同时，将渗透技术应用于负载均衡。由于渗透技术支持通过云基础设施迁移的虚拟机的自动部署，从而克服了现有仿生算法在实现物理机之间负载平衡方面的缺点，提高了迁移效率。实验结果表明，以现有负载平衡算法相比，提出的算法在迁移性能上明显提升。

关键词：云计算；负载平衡；渗透技术；人工蜂群；蚁群优化

中图分类号：TP393 doi: 10.19734/j.issn.1001-3695.2019.11.0658

Hybrid load balancing algorithm based on osmotic artificial bee colony and ant colony optimization

Hou Xiang, Yang Chengfu, Liu Dujin

(*School of intelligent manufacturing*, *Sichuan University of Arts & Science*, *Dazhou* 635000, *China*)

Abstract: Aiming at the low efficiency and high energy consumption of virtual machine migration in the current cloud computing load balancing scheduling process, a hybrid load balancing algorithm based on osmotic artificial bee colony and ant colony is proposed. The algorithm combines chemical permeation behavior with bio-inspired load balancing algorithm to apply the osmotic technique to load balancing while making full use of the advantages of both artificial bee colony and ant colony optimization algorithms. Because the osmotic technology supports the automatic deployment of virtual machines migrated through the cloud infrastructure, it overcomes the shortcomings of the existing bionic algorithms in realizing the load balancing between physical machines and improves the migration efficiency. The experimental results show that compared with the existing load balancing algorithm, the proposed algorithm has significantly improved the migration performance.

Key words: cloud computing; load balancing; osmotic technique; artificial bee colony; ant colony optimization

0 引言

根据美国国家标准技术研究院(NIST)的定义，云计算被定义为“按使用付费模式”，它可以让可用的、方便的、按需的网络访问共享的可配置计算资源池(如网络、服务器、存储、应用程序、服务)[1]。由于资源共享和使用需求的快速增长，云计算在用户数量不断增加的情况下面临着诸多挑战。因此，资源之间的负载平衡是一个重要的挑战[2, 3]。

云环境中的资源负载平衡是指通过算法将用户任务请求平均分配到服务器上来提高资源利用率的技术[4]。云计算的核心是资源池中资源的分配问题，云计算的调度就是实现数据中心内虚拟器之间的负载平衡。目前，许多云计算的负载平衡是基于启发式优化算法提出了[5]。Sharma 等[6]提出了一种基于负载均衡增强遗传算法的云任务调度策略，在尽量减少给定任务集的有效时间跨度的同时，保持整个系统负载的平衡。Sajjan 等[7]提出了一种基于任务的云环境负载均衡方法，该方法使用 k-means 聚类方法将虚拟机分组，然后采用遗传算法、模拟退火和粒子群优化三种启发式算法来降低完成时间。Dam 等[8]使用蚁群优化算法来平衡云计算中虚拟机的负载，该算法通过将动态工作负载平均分配到整个系统中来平衡负载并优化响应时间。Kong 等[9]提出了一种基于零不平衡方法的快速启发式算法，该算法侧重于最大程度地减少异构虚拟机之间的完成时间差异，而无须优先级方法和复杂的调度决策，从而有效地实现了负载平衡和任务调度。Shen 等[10]提出了一种基于负载均衡算法的人工蜂群优化问题，以提高系统的整体负载均衡性能，获得更好的自适应性。Gamal 等[11]提出了一种混合人工蜂群和蚁群优化的负载平衡算法，它依赖于蚁群优化和人工蜂群算法提高了系统的执行时间和资源利用率。

尽管当前许多启发算法在负载平衡系统中都具有明显的有效性，然而大多数算法的缺点也很突出，如人工蜂群算法在顺序处理中计算效率不高，而蚁群优化的收敛速度较慢。因此，本文提出了一种渗透式人工蜂群(artificial bee colony，ABC)与蚁群(ant colony optimization，ACO)混合优化负载平衡算法，在充分利用人工蜂群和蚁群优化两种算法优点的同时，将渗透技术应用于负载均衡。

1 渗透技术

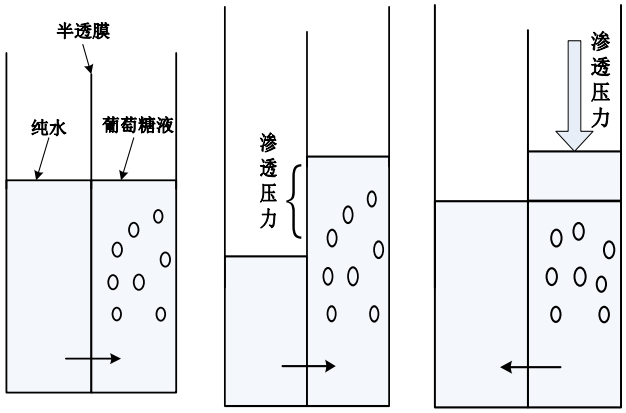
在化学中，渗透表示分子从高浓度向低浓度无限制净运动的过程。如图 1 所示，当纯净水和葡萄糖液通过半透膜分开时，水从高水活度向低水活度移动[12]。为了保持平衡，通过在溶液中施加渗透压以阻止水流过半透膜，数学公式可以表示为

$$\pi = ic_{solute}RT \tag{1}$$

收稿日期：2019-11-29；**修回日期：**2020-03-02 **基金项目：**四川省哲学社会科学重点研究基地、四川省教育厅人文社会科学重点研究基地——四川革命老区发展研究中心资助项目：川陕革命老区优势农业暨特色农产品大数据系统研究及云平台开发(SLQ2019B-26)

作者简介：侯翔(1983-)，男，四川达州人，副教授，硕士，主要研究方向为智能计算、人工智能(dzhouxiang@163.com)；杨成福(1972-)，男，四川达县人，副教授，博士，主要研究方向为人工智能、语音信号处理；刘笃晋(1971-)，男，四川渠县人，副教授，博士，主要研究方向为数字图像处理，机器学习.

其中，π 表示渗透压，i 表示校正系数，c_{solute} 表示溶液摩尔浓度，R 为理想气体常数，T 为开尔文温度。



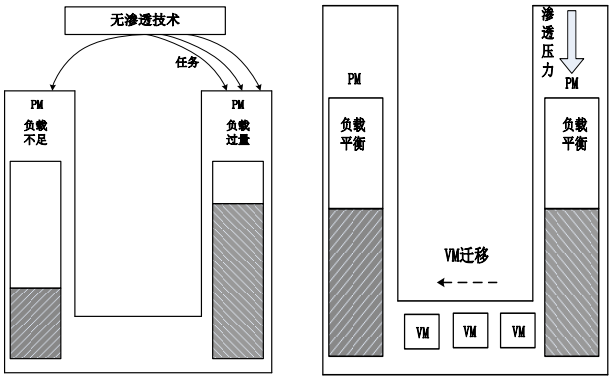
(a)初始状态　(b)平衡状态　(c)外压状态

(a) Initial state　(b) balanced state　(c) external pressure state

图 1 渗透技术示意图

Fig. 1　Schematic diagram of infiltration technology

在云环境中，该过程可用于表示虚拟机(virtual machine，*VM*)在云计算之间的迁移过程，如图 2 所示。图 2(a)显示了物理机(physical machine，*PM*)负载过高和负载过少的两种液体，分别定义为纯净水和葡萄糖液。图 2(b)显示了渗透技术的工作原理，通过在 *PM* 之间迁移 *VM*，实现更加平衡的云系统。



(a)不平衡系统　(b)平衡系统

图 2 渗透技术在负载均衡中的作用示意图

Fig. 2　Schematic diagram of the role of infiltration technology in load balancing

2　渗透式混合负载平衡算法

近年来，科研人员趋向于应用化学中的渗透理论来形成渗透计算。本文的目标是根据渗透原理来实现负载均衡。此外，还将 ACO 和 ABC 与渗透技术结合起来，在克服缺点的同时利用它们的优势。云环境中的活动主机分为负载不足，过载和平衡的主机，本文算法通过监视系统发现活动主机的状态，然后将 *VM* 从过载主机上迁移到负载不足的主机，以实现负载均衡的系统。提出的算法继承了 ACO 在多样性系统中快速发现解决方案的优点和 ABC 通过建立知识库共享信息的行为。本文算法应用具有渗透技术的知识库来根据能耗分类 *PM*，避免了 ACO 随机选择 *PM* 时产生收敛过慢的现象。同时，还将根据云系统的状态考虑阈值的动态值。图 3 给出了提出算法的流程示意图。

在云计算环境中，每个 *PM* 都有不同数量的 *VM*。数据中心中所有 *PM* 的集合表示为 $P=\{P_1,P_2,\ldots,P_n\}$，数据中心中所有 *VM* 的集合是 $VM=\{V_1,V_2,\ldots,V_m\}$，其中 n、m 表示物理机和虚拟机的数量。在提出的算法中，侦查蜂负责计算每个 *PM* 的标准差 σ，目标是找到未充分利用和过度利用的主机。这需要找到每个 *PM* 的负载，具体取决于部署到其中的 *VM* 的负载。第 i 个 *PM* 中的第 j 个 *VM* 平均负载 $\bar{V}_{ij}$ 计算如下：

$$\bar{V}_{ij}=U_{Cpu_j}+U_{Mem_j}+U_{Bw_j} \tag{2}$$

其中，U_{Cpu_j}、U_{Mem_j} 和 U_{Bw_j} 分别表示第 j 个虚拟机 VM_j 的 CPU 利用率，内存利用率和带宽利用率。

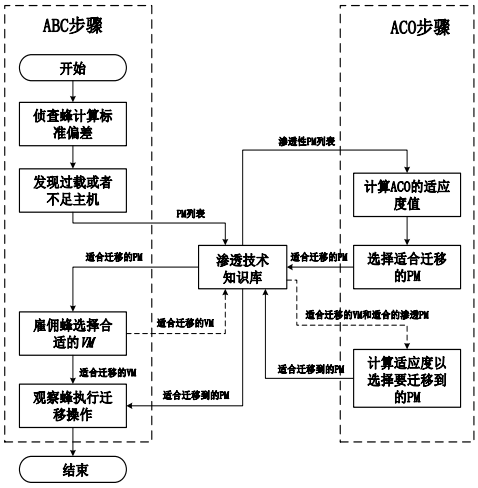


图 3　提出算法的流程示意图

Fig. 3　Flow chart of the proposed algorithm

PM_i 的平均负载 $\bar{P}_i$ 和标准偏差 σ_i 计算如下：

$$\bar{P}_i=\frac{\sum_{j=1}^{m}\bar{V}_j}{m},\quad \forall V_{1,2,\ldots,m}\in P_i \tag{3}$$

$$\sigma_i=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{P}_T-\bar{P}_i)^2} \tag{4}$$

$$\bar{P}_T=\frac{1}{n}\sum_{i=1}^{n}\bar{P}_i \tag{5}$$

如果 σ_i 小于下限阈值，则 PM_i 未得到充分利用；如果 σ_i 超过上限阈值，则 PM_i 超出主机利用率。阈值计算如下：下限阈值设定为所有 *PM* 中的最小 $\bar{P}_i$，上限阈值设定为 $\bar{P}_T$。

找到利用不足或者过度利用的主机后，人工蜂群中的侦查蜂会在知识库中摆动，以告知所有蜂群有关其结果的信息。然后，知识库通过(6)式利用渗透技术来安排主机，并考虑每个 *PM* 的功耗，如图 3 所示。

$$\pi PM_n=iPC_{PM_n}N_{PMs}L_{PM_n} \tag{6}$$

其中，πPM_n 表示渗透压，i 表示校正系数，PC_{PM_n} 表示 PM_n 的能耗，N_{PMs} 表示云系统中 *PM* 数量的常数，L_{PM_n} 表示 PM_n 的负载。

然后，将新的主机列表发送到 ACO，ACO 根据列表开始寻找所有渗透主机中合适的 *PM*，以从中进行虚拟机迁移。ACO 根据 PM 的在利用率方面的高低分类计算其适应性：

$$F_i=\frac{(\tau_i)^{\alpha}*(\eta_i)^{\beta}}{\sum_{i=1}^{m}(\tau_i)^{\alpha}*(\eta_i)^{\beta}} \tag{7}$$

$$F_i=\frac{(1/\tau_i)^{\alpha}*(\eta_i)^{\beta}}{\sum_{i=1}^{m}(1/\tau_i)^{\alpha}*(\eta_i)^{\beta}} \tag{8}$$

其中，α、β 表示信息素 τ_i 和边缘权重 η_i 的系数，信息素 τ_i 表示 PM_i 的负载，η_i 表示带宽。

通过选择最合适的 PM_i 使其处于迁移状态并利用知识库通知其他蚁群。然后更新信息素：

$$\tau_i=(1-\rho)\tau_i+\Delta\tau_i \tag{9}$$

$$\Delta\tau_i=\frac{1}{(\eta_i)^{\gamma}}+P_i \tag{10}$$

其中，ρ 表示挥发速率，P_i 是 *PM* 的负载，γ 是用于定义启发式值与负载条件比例的参数。

之后，雇佣蜂执行计算，以通过最短迁移时间策略选择要迁移到另一台主机的合适 *VM*，该策略是 *VM* 使用的 RAM 数量除以主机可用的备用网络带宽：

$$\frac{RAM(a)}{net_i} \le \frac{RAM(u)}{net_i},\ \forall a,u \in V_j \tag{11}$$

其中，V_j 是当前分配给主机 P_i 的一组 *VM*，net_i 是 P_i 可用的备用网络带宽，$RAM(a)$ 和 $RAM(u)$ 分别表示 VM_a 和 VM_u 当前使用的 RAM 量。

随后，ACO 计算适应度值，找到所选 *VM* 与相匹配渗透主机 *PM* 之间的最佳映射关系：

$$Fitness(VM_j, PM_i) = \frac{PM_{Cpu_i} - VM_{Cpu_j}}{VM_{Cpu_j}} \cdot \frac{PM_{Mem_i} - VM_{Mem_j}}{VM_{Mem_j}} \cdot \frac{PM_{Net_i} - VM_{Net_j}}{VM_{Net_j}} \cdot \frac{PM_{Storage_i} - VM_{Storage_j}}{VM_{Storage_j}} \tag{12}$$

其中，VM_{Cpu_j}、VM_{Mem_j}、VM_{Net_j} 和 $VM_{Storage_j}$ 分别表示 *VM* 所需的 CPU 利用率、内存、带宽和存储大小，PM_{Cpu_i}、PM_{Mem_j}、PM_{Net_j} 和 $PM_{Storage_j}$ 分别表示 *PM* 具备的 CPU 利用率、内存、带宽和存储大小。最后，观察蜂获取有关 *VM* 的信息，根据这些信息将 *VM* 迁移至合适的 *PM*。

3　实验结果与分析

3.1　实验环境

为了验证所提出算法的性能，在 CloudSim API 3.0.3 实现。模拟环境中包含 50 个 PM，其中 50％的主机是 HP ProLiant ML110 G4(Intel Xeon 3040，2 核@1860 MHz，RAM4GB)，另一种是 HP ProLiant ML110 G5(Intel Xeon 3075，2 核@2660 MHz，RAM 4GB)。每个主机都有 1GB / s 的网络带宽。考虑数据中心中具有 50 个异构 VM，所有 VM 的类型均为单核，RAM 会根据 VM 的类型进行划分：超大型实例(2000 MIPS，3.75 GB)、高 CPU 中型实例(2500 MIPS，0.85 GB)、小型实例(1000 MIPS，1.7 GB)以及微型实例(500 MIPS，613 MB)。表 1 给出了算法的其他参数。

表 1　提出算法的相关参数

Tab. 1　Relevant parameters of the proposed algorithm

参数	值	参数	值
迭代次数	100	边缘权重系数 β	0.32
蚁群数量	5	负载比例参数 γ	0.8
蜂群数量	15	挥发速率 ρ	0.1
信息素系数 α	0.8	校正系数 i	0.8

3.2　结果分析

本文将通过两次实验将提出算法与现有方案进行比较。首先，将本文算法与 ACO [8]，ABC [10]，H_BAC [11]和指数加权移动平均(EWMA)[13]以及多维回归主机利用率算法(MDRHU)[14]进行对比，验证算法对主机过载检测的性能。其次，将算法与与 ACO [8]，ABC [10]，H_BAC [11]的可变任务负载测试结果进行比较。

3.2.1 主机过载检测

在第一个实验中，提出算法与其他算法在能耗、SLA 违规(SLAV)、每个活动主机的 SLA 冲突时间(SLATAH)、迁移导致性能下降(PDM)、主机关闭数和虚拟机迁移数等几个方面进行比较。该实验适用于固定参数，任务数、主机和 VM 的数量均等于 50。

在作出迁移决策时考虑 SLA。当发生过载和欠载情况时，PM 按其负载进行排序，以便根据 SLA 在其容量合适时检测用于 *VM* 迁移的合适 *PM*。SLAV 是一个有用的度量，用于评估 IaaS 云中 VM 交付的 SLA。SLATAH 是活动主机 100%使用 CPU 的时间百分比，PDM 是由于迁移而导致的性能下降的评价指标。三个指标的数学公式定义如下：

$$SLATAH = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{si}}{T_{ai}} \tag{13}$$

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{dj}}{C_{rj}} \tag{14}$$

$$SLAV = SLATAH \times PDM \tag{15}$$

其中，N、M 分别表示 *PM* 和 *VM* 的数量，T_{si} 表示第 i 个 *PM* 被 100％利用的时间，T_{ai} 是处于活动状态的第 i 个 *PM* 的总数，C_{dj} 表示由于迁移导致的第 j 个 *VM* 性能下降的评估，C_{rj} 表示第 j 个 *VM* 所需的总体 CPU 容量。

表 2 给出了所提出的算法与其他算法进行比较的仿真结果。结果表明，H_BAC 算法比 ACO 和 ABC 算法消耗更多的能量，而本文算法通过考虑每个 PM 的功耗并且选择最低功耗的 PM，使得在所有算法中能量消耗最低。同时，提出的算法增强了 PDM，因为通过混合 ACO 和 ABC 算法可以选择最佳 VM，以迁移到最合适的 PM。但是，本文算法具有更高的 SLATAH，根据式(13)的结果，该结果取决于活动主机的数量，与其他确定活动主机然后关闭的主机的算法相比，本文使活动主机的数量最小化。总体来说，与其他算法相比，本文算法提高了能耗、SLAV、VM 迁移数和主机关闭数。但是，该算法具有更高的 SLATAH，但对云系统的性能没有影响。

表 2　不同主机过载检测算法的测试结果对比

Tab. 2　Comparison of test results of different host overload detection algorithms

算法	能耗	SLAV	SLATAH	PDM	PM 关闭数量	VM 迁移数量
ABC	28.6	33.5	32	0.11	449	1404
ACO	34.4	46	27	0.17	781	2355
H_BAC	40	28	26	0.11	525	1532
EWMA	46	19	9	0.23	1549	2871
MDRHU	48	17	8	0.23	1528	2502
本文算法	20	2	55	0.01	46	81

3.2.2 任务负载调度

在第二个实验中，随着任务数量从 50 个逐渐增加到 250 个，任务的负载是可变的。图 4~9 给出了能耗、SLAV、PDM、SLATAH、主机关闭数和虚拟机迁移数的测试结果。

图 4 给出了不同算法在能耗方面的测试结果。结果表明，与 H_BAC 算法相比，本文算法的性能提高了约 27%，与 ABC 算法相比提高了 21%，与 ACO 算法相比提高了 18%。

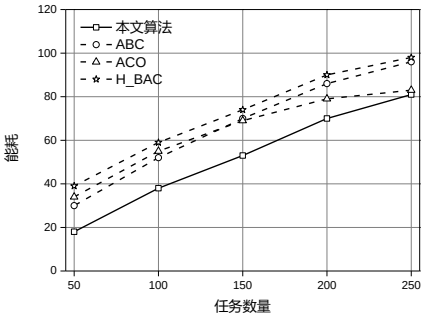


图 4　不同算法在能耗方面的对比

Fig. 4　Comparison of different algorithms in terms of energy consumption

从图 5、6 分别给出了不同算法在 SLAV 和 PDM 的测试结果。从图中可以看出，提出算法的优势明显，充分利用了 ACO 和 ABC 优化算法的优点，并且改善了 H_BAC 混合算法的性能。

与 H_BAC、ABC 和 ACO 算法相比，提出算法的 SLATAH 分别多了 24%、34%和 30%，如图 7 所示。但是，它不会影响云系统的性能。根据方程式(13)的结果，算法将活动主机的数量分别减少了 93%、91%和 94%，如图 8 所示。关闭的主机数与云环境中的主机数无关。如果此主机没有任何任务，则该主机将关闭，此过程可以在虚拟机迁移后完成。如果一

个主机中的所有虚拟机都已迁移，则该主机将关闭以降低功耗。但是，如果有一个 VM 再次迁移到主机，则该主机可能会被激活。在算法中，如果有任何迁移，首先检查活动主机，然后如果所有活动主机都无法接受此虚拟机，则它将检查未活动主机并打开一个。

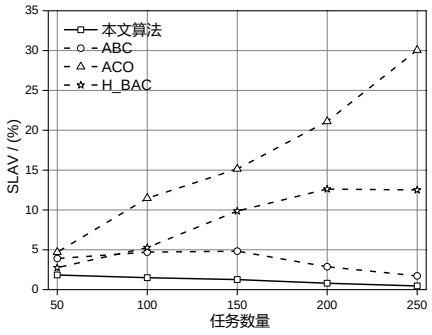


图 5　不同算法在 SLAV 方面的对比

Fig. 5　Comparison of different algorithms in SLAV

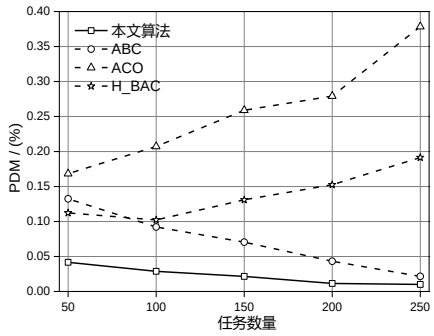


图 6　不同算法在 PDM 方面的对比

Fig. 6　Comparison of different algorithms in PDM

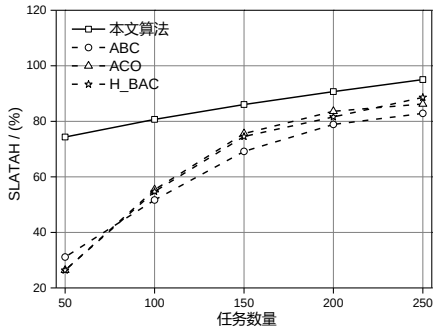


图 7　不同算法在 SLATAH 方面的对比

Fig. 7　Comparison of different algorithms in SLATAH

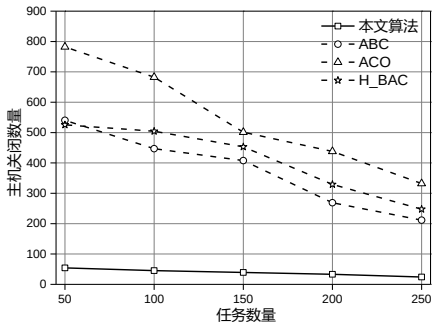


图 8　不同算法在主机关闭方面的对比

Fig. 8　Comparison of different algorithms in host shutdown

实时虚拟机迁移是一个代价高昂的过程，它在源 *PM* 上包含一定数量的 CPU 处理。此外，它还取决于源 *PM* 和目标 *PM* 之间的带宽。由于迁移 *VM* 进程会增加任务的响应时间，所以本文目标是最小化 *VM* 迁移数量。因此，*VM* 迁移数量被认为是一个度量标准，用来评估算法的性能。如图 9 所示，在其他算法中，所提算法的 *VM* 迁移数量最少。

4　结束语

本文提出了一种基于渗透式人工蜂群与蚁群混合优化负载平衡算法，用于解决当前云计算负载平衡调度过程中出现的虚拟机迁移效率低和能耗高问题。该算法在继承了 ACO 和 ABC 两种仿生优化算法优势的基础上，引入了渗透技术，克服了两种算法在实现物理机之间负载平衡方面的缺点，有效提升了平衡算法的性能。实验结果表明，在固定和可变负载的两个实验中，提出的算法均具有明显的优势。

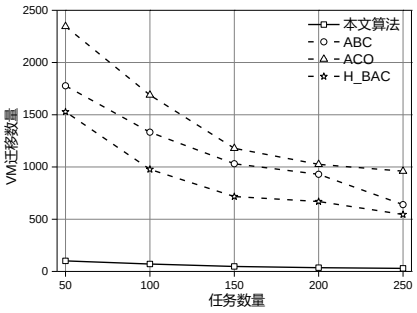


图 9　不同算法在 VM 迁移方面的对比

Fig. 9　Comparison of different algorithms in VM migration

参考文献：

[1] Nashaat H, Ashry N, Rizk R. Smart elastic scheduling algorithm for virtual machine migration in cloud computing [J]. The Journal of Supercomputing, 2019, 75 (7): 3842-3865.

[2] Xu Xiaolong, Liu Qingxiang, Qi Lianyong, *et al.* A Heuristic Virtual Machine Scheduling Method for Load Balancing in Fog-Cloud Computing [C]// 2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity) , IEEE International Conference on High Performance and Smart Computing (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS) . Omaha, USA: IEEE, 2018: 83-88.

[3] Ashouraei M, Khezr S N, Benlamri R, *et al.* A new SLA-aware load balancing method in the cloud using an improved parallel task scheduling algorithm [C]// 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud) . Barcelona, Spain: IEEE, 2018: 71-76.

[4] Yang Zhexi, Zhang Suyin, Ji Xiaoye, *et al.* Research on cloud service quality control implementation based on improved load balance algorithm [J]. Journal of Computational Methods in Sciences and Engineering, 2018, 18 (3): 793-800.

[5] 王红运，束永安. 数据中心网络中基于蚁群算法的动态多路径负载均衡 [J]. 计算机应用研究. 2020, 37 (07) . Wang Hongyu, Su Yongan. Dynamic multi-path load balancing based on Ant Colony Algorithm in data center network [J]. Application Research of Computers. 2020, 37 (07)

[6] Sharma H, Sekhon G S. Load Balancing in Cloud Using Enhanced Genetic Algorithm [J]. 2017, 6 (1): 13-19.

[7] Sajjan R S, Yashwantrao B R. Load Balancing using Cluster and Heuristic Algorithms in Cloud Domain [J]. Indian Journal of Science and Technology, 2018, 11 (15): 1-7.

[8] Dam S, Mandal G, Dasgupta K, *et al.* An ant-colony-based meta-heuristic approach for load balancing in cloud computing [M]// Applied Computational Intelligence and Soft Computing in Engineering. IGI Global, 2018: 204-232.

[9] Kong Lingfu, Mapetu J P B, Chen Zhen. Heuristic Load Balancing Based

Zero Imbalance Mechanism in Cloud Computing [J]. Journal of Grid Computing, 2019, 7: 1-26.

[10] Shen Luocheng, Li Jiazhou, Wu Yan, *et al.* Optimization of Artificial Bee Colony Algorithm Based Load Balancing in Smart Grid Cloud [C]// 2019 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia) . Chengdu, China: IEEE, 2019: 1131-1134.

[11] Gamal M, Rizk R, Mahdi H, *et al.* Bio-inspired Based Task Scheduling in Cloud Computing [M]// Machine Learning Paradigms: Theory and Application. Springer, Cham, 2019: 289-308.

[12] Villari M, Celesti A, Fazio M. Towards osmotic computing: Looking at basic principles and technologies [C]// 2017 Conference on Complex, Intelligent, and Software Intensive Systems (CISIS) . Turin, Italy: Springer, Cham, 2017: 906-915.

[13] Lu Shinli, Chen Jenhsiang. Host Overloading Detection Based on EWMA Algorithm in Cloud Computing Environment [C]// 2018 IEEE 15th International Conference on e-Business Engineering (ICEBE) . Xi'an, China: IEEE, 2018: 274-279.

[14] 谢兵. 基于移动云计算的计算迁移能效算法 [J]. 计算机应用研究. 2020, 37 (10) . Xie Bing. Energy efficiency algorithm of computing migration based on mobile cloud computing [J]. Application Research of Computers. 2020, 37 (10)