软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(11):3944−3966 [doi: 10.13328/j.cnki.jos.006330]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于混合群智能的节能虚拟机整合方法*

李俊祺[1], 林伟伟[1,3], 石 方[1], 李克勤[2]

[1](华南理工大学 计算机科学与工程学院, 广东 广州 510006)

[2](Department of Computer Science, State University of New York, NY 12561, USA)

[3](鹏程实验室, 广东 深圳 518066)

通信作者: 林伟伟, E-mail: linww@scut.edu.cn

摘 要: 数据中心的虚拟机(virtual machine, VM)整合技术是当今云计算领域的一个研究热点. 要在保证服务质量(QoS)的前提下尽可能地降低云数据中心的服务器能耗, 本质上是一个多目标优化的 NP 难问题. 为了更好地解决该问题, 面向异构服务器云环境提出了一种基于差分进化与粒子群优化的混合群智能节能虚拟机整合方法(HSI-VMC). 该方法包括基于峰值效能比的静态阈值超载服务器检测策略(PEBST)、基于迁移价值比的待迁移虚拟机选择策略(MRB)、目标服务器选择策略、混合离散化启发式差分进化粒子群优化虚拟机放置算法(HDH-DEPSO)以及基于负载均值的欠载服务器处理策略(AVG). 其中, PEBST, MRB, AVG 策略的结合能够根据服务器的峰值效能比和 CPU 的负载均值检测出超载和欠载服务器, 并选出合适的虚拟机进行迁移, 降低负载波动引起的服务水平协议违约率(SLAV)和虚拟机迁移的次数; HDH-DEPSO 算法结合 DE 和 PSO 的优点, 能够搜索出更优的虚拟机放置方案, 使服务器尽可能地保持在峰值效能比下运行, 降低服务器的能耗开销. 基于真实云环境数据集(PlanetLab/Mix/Gan)的一系列实验结果表明: HSI-VMC 方法与当前主流的几种节能虚拟机整合方法相比, 能够更好地兼顾多个 QoS 指标, 并有效地降低云数据中心的服务器能耗开销.

关键词: 云计算; 虚拟机整合; 差分进化算法; 粒子群优化算法; 节能

中图法分类号: TP311

中文引用格式: 李俊祺, 林伟伟, 石方, 李克勤. 基于混合群智能的节能虚拟机整合方法. 软件学报, 2022, 33(11): 3944−3966. http://www.jos.org.cn/1000-9825/6330.htm

英文引用格式: Li JQ, Lin WW, Shi F, Li KQ. Energy Efficient Hybrid Swarm Intelligence Virtual Machine Consolidation Method. Ruan Jian Xue Bao/Journal of Software, 2022, 33(11): 3944−3966 (in Chinese). http://www.jos.org.cn/1000-9825/6330.htm

Energy Efficient Hybrid Swarm Intelligence Virtual Machine Consolidation Method

LI Jun-Qi[1], LIN Wei-Wei[1,3], SHI Fang[1], LI Ke-Qin[2]

[1](School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China)

[2](Department of Computer Science, State University of New York, NY 12561, USA)

[3](Peng Cheng Laboratory, Shenzhen 518066, China)

Abstract: Virtual machine (VM) consolidation for cloud data centers is one of the hottest research topics in cloud computing. It is challenging to minimize the energy consumption while ensuring QoS of the hosts in cloud data centers, which is essentially an NP-hard multi-objective optimization problem. This study proposes an energy efficient hybrid swarm intelligence virtual machine consolidation method (HSI-VMC) for heterogeneous cloud environments to address this issue, which including peak efficiency based static threshold overloaded hosts detection strategy (PEBST), migration ratio based reallocate virtual machine selection strategy (MRB), target host

* 基金项目: 广东省重点领域研发计划(2020B010164003); 国家自然科学基金(62072187, 61872084); 广东省基础与应用基础研究基金(2019B030302002); 广州市科学研究计划(202007040002, 201902010040, 201907010001); 广州开发区科技项目(2020GH10)

收稿时间: 2020-10-02; 修改时间: 2020-12-15, 2021-02-03; 采用时间: 2021-03-03; jos 在线出版时间: 2021-08-03

selection strategy, hybrid discrete heuristic differential evolutionary particle swarm optimization virtual machine placement algorithm (HDH-DEPSO) and load average based underloaded hosts processing strategy (AVG). Specifically, the combination of PEBST, MRB, and AVG is able to detect the overloaded and underloaded hosts and selects appropriate virtual machines for migration to reduce SLAV and virtual machine migrations. Also, HDH-DEPSO combines the advantages of DE and PSO to search the best virtual machine placement solution, which can reduce cluster's real-time power effectively. A series of experiments based on real cloud environment datasets (PlanetLab, Mix, and Gan) show that HSI-VMC can reduce energy consumption sharply with accommodate to multiple QoS metrics, outperforms several existing mainstream energy-aware virtual machine consolidation approaches.

Key words: cloud computing; virtual machine consolidation; differential evolution algorithm; particle swarm optimization algorithm; energy efficient

有研究表明: 预计到 2020 年底, 全球的超大规模数据中心数量将突破到 500 以上[1], 其数据量在未来几年的年均增幅将高达 40%[2]. 数据中心高速扩张所带来的能源消耗和环境保护问题逐渐引起人们的重视. 典型数据中心的功率密度为 558 W/m^2–2 153 W/m^2, 有时甚至达到 10 kw/m^2, 其主要能耗来源于 IT 设备、制冷设备、光照、配电等因素[3]. 服务器节能是当前数据中心节能问题的一个重要改良点[4]. 现有的数据中心服务器节能技术主要分为动态电压频率调整技术(DVFS)和虚拟机整合技术这两种[4−6]: 动态电压频率调整技术是一种动态服务器调控技术, 它能根据处理器、内存、高速缓存等多个组件的使用情况和应用程序对不同计算资源的需求, 灵活地调整服务器的电压和频率, 在保证用户 QoS 的同时实现服务器集群的整体节能; 虚拟机整合技术是一种面向虚拟机的计算资源分配技术, 它通过虚拟机的放置和迁移调整各服务器的负载水平, 并尝试对欠载服务器进行动态关断以减少活跃服务器的数量, 其目标是寻找一种合适的"虚拟机-服务器"映射关系, 将虚拟机整合到有限的物理资源子集上, 以减少集群功耗与资源碎片, 提升服务器的能源效率并促进集群的负载平衡程度, 在最大程度上保证服务质量(QoS)与资源的有效利用. 动态电压频率调整技术和虚拟机整合技术能够相互融合, 根据用户的资源需求动态调整服务器的功率和负载状况, 因而被广泛应用于数据中心的节能管理当中[7]. 然而, 虚拟机整合具有多个相互矛盾的优化目标, 数据中心庞大的集群规模导致虚拟机整合算法难以在多项式时间内获取到最优的调度方案. 因此, 该问题是一个面向多目标优化的 NP 难问题.

为获取较优的虚拟机调度方案, 并进一步降低服务器的功耗, 很多学者使用启发式算法来解决虚拟机整合问题[8,9]. 传统启发式算法由直观概念或经验构成, 包括首次适应算法(FFD)[10]、轮询调度算法(round-robin)、Max-Min 算法、Min-Min 算法[11]、最佳适应递减算法(BFD)[12]等. 它们希望通过某种启发式规则, 在可接受的时间和空间限制下给出问题的可行解. 其实现过程简单, 运行时间较短, 适用于虚拟机的动态调度场景. 然而该类算法得到的解往往是问题的局部最优解, 因而在解决虚拟机的节能调度问题中存在限制. 群智能算法通过模拟群体的社会行为搜索问题的最优解, 是一类高效的启发式算法. 常见的群智能算法包括遗传算法(GA)[13]、差分进化算法(DE)[14]、粒子群优化算法(PSO)[15]、蚁群优化算法(ACO)[16]、人工蜂群算法(ABC)[17]等. 它们结合迭代过程中个体和群体的信息对问题的可行解进行搜索, 与传统的启发式算法相比能够得到更优的解, 在虚拟机调度、系统异常检测[18]、网络资源分配[19]、神经网络参数调整[20]、视频目标跟踪[21]、智慧城市行程推荐[22]与能源管控[23]等多个领域得到广泛的应用. 然而在虚拟机整合问题中, 群智能算法需要进行多次迭代来求解问题, 其迭代过程可能出现早熟收敛、陷入局部最优等问题, 因此仍存在一定的改良空间.

现有的主流虚拟机整合策略为达到降低能耗的目的, 易导致服务水平协议违约率(SLAV)和虚拟机迁移次数的激增, 影响数据中心的 QoS. 为此, 本文提出了一种新型的虚拟机整合方法(HSI-VMC). 该方法根据服务器的峰值效能比确定各服务器的负载上限阈值, 并通过虚拟机迁移价值比从超载服务器中选出合适的虚拟机重新放置, 能够大幅度减少虚拟机迁移的次数并保证数据中心的 QoS. 同时, 一种融合了 MBFD, DE 和 PSO 的混合离散化启发式差分进化粒子群优化(HDH-DEPSO)算法被应用于 HSI-VMC 的虚拟机放置阶段. 利用 DE 的交叉变异操作对迭代过程中的虚拟机放置方案进行扰动, 从而避免 PSO 在迭代后期陷入局部最优, 帮助算法找到更优的虚拟机放置方案. 本文的主要贡献包括:

(1) 本文面向异构服务器集群环境提出了一种基于混合群智能的节能虚拟机整合方法(HSI-VMC), 该方法包括基于峰值效能比的静态阈值(PEBST)超载服务器检测策略、基于迁移价值比的待迁移虚拟机

选择策略(MRB)、目标服务器选择策略、混合离散化启发式差分进化粒子群优化虚拟机放置算法(HDH-DEPSO)、基于负载均值的欠载服务器处理策略(AVG). HSI-VMC 能够保持服务器的高效运行，在兼顾多个 QoS 指标的同时，有效地降低服务器集群的能耗.

(2) 针对虚拟机整合过程中的超载欠载服务器检测问题与虚拟机放置问题，我们分别提出了基于峰值效能比的静态阈值超载服务器检测策略(PEBST)、基于负载均值的欠载服务器处理策略(AVG)以及混合离散化启发式差分进化粒子群优化虚拟机放置算法(HDH-DEPSO). 其中，PEBST 策略能够恰当地检测出超载服务器，并进一步通过 MRB 策略选出尽可能少的虚拟机进行迁移，减少虚拟机迁移次数；AVG 策略能够根据服务器的 CPU 利用率选出空闲服务器进行关断，减少活跃服务器的数量；HDH-DEPSO 重新定义了 DE 和 PSO 的离散化运算规则，并通过贪心启发式算法 MBFD 解决了迭代过程中的虚拟机放置冲突问题，能够给出节能高效的虚拟机放置方案.

(3) 我们通过实验进一步分析了 HSI-VMC 在不同规模集群下能耗、SLAV、虚拟机迁移次数等评价指标上的表现. 实验结果表明：HSI-VMC 能够在保持较低 SLAV 的同时，有效地降低集群能耗，得到更优的虚拟机调度方案.

本文第 1 节介绍近年来群智能虚拟机调度算法的研究现状. 第 2 节提供虚拟机整合问题的相关定义，并明确问题的优化目标. 第 3 节详细介绍 HSI-VMC 的整体流程及整合过程中各阶段的相关设计. 第 4 节阐述 HSI-VMC 在 CloudSim 4.0[24]平台异构服务器环境下仿真实验的表现，并从多个方面与现有的主流虚拟机整合方法做比较. 第 5 节对本文的研究进行总结并给出未来的研究方向.

1 相关研究

虚拟机调度是一个多目标优化问题，如何在多峰值的搜索空间中寻找高效的调度方案，是虚拟机整合算法的研究重点. 群智能算法能够通过多次迭代，在多个目标之间取得平衡，以获取较优的解，在虚拟机调度问题中有着优异的表现.

遗传算法和差分进化算法模仿基因的交叉和变异行为，对初始群体中的解向量进行大幅度扰动，进而搜索出满足需求的调度解，具有较强的全局搜索能力. 文献[25]提出了一种基于遗传算法的节能虚拟机整合算法(ECVMC)，并给出了解向量的分段编码方式. ECVMC 利用服务器的 CPU、内存、磁盘利用率估算出解的质量，能够在减少服务器能耗的同时，尽可能地降低用户的支出. Li 等人为了兼顾服务器的能耗开销和用户的 QoS，提出了一种基于差分进化算法的虚拟机整合方法 EQ-VMC[26]. EQ-VMC 利用一种正态分布模型对服务器资源的使用记录进行拟合，获取不同资源的概率分布函数，并通过概率分布函数计算服务器超载概率和负载上限下限阈值. 同时，作者重新定义了差分进化算法的变异、交叉、选择操作，使 EQ-VMC 能够获取兼顾能耗和 QoS 的较优调度解. Jiang 等人对传统的遗传算法的编码方式进行了改进，并提出了 VMM-GAGA 虚拟机放置算法[27]. 作者对虚拟机放置算法进行优化，采用双点交叉和基本位变异操作，降低了关联虚拟机之间的通信代价，并缩减了算法迭代过程的运行时间.

粒子群优化算法模仿鸟群的迁徙行为，利用个体最优解和群体最优解引导算法的搜索，能够通过多次迭代获取到较优的虚拟机调度方案. Wang 等人给出了一种双维度的解向量编码和离散向量的速度位置更新方法，并基于该方法提出了一种解决异构服务器环境虚拟机放置问题的离散化粒子群优化算法[28]. 该算法能够结合能耗感知的本地适应优先调度策略，获取能耗开销较低的虚拟机调度方案. Ma 等人提出了一种惯性因子和学习因子动态变化的 SE-PSO 虚拟机调度算法[29]. SE-PSO 基于一种指数平滑模型的超载服务器预测方法和虚拟机选择方法协助粒子群优化算法对虚拟机进行调度，减少了集群的能耗开销和虚拟机迁移次数. Yan 等人提出了一种离散化的粒子群虚拟机调度优化算法 SOWO[30]，有效减少了集群的活跃服务器数量和资源碎片. Sharma 等人提出了一种基于 HGAPSO 混合群智能算法的虚拟机调度策略，用于解决数据中心的资源分配和虚拟机调度问题[31]. 该算法融合了遗传算法和粒子群优化算法，缓解了粒子群优化算法迭代过程易陷入局部最优的问题，在数据中心服务器节能方面有着不错的表现. Meshkati 等人针对虚拟机放置问题，给出了

HSF.ABC&PSO 虚拟机放置算法[32]. 该放置算法通过人工蜂群算法和粒子群算法的并行化, 加强了人工蜂群算法的迭代收敛速度, 并弥补了粒子群优化算法迭代后期欠缺全局搜索能力的缺点.

蚁群算法模仿蚁群的觅食行为, 通过信息素积累和蚂蚁的路径选择策略生成虚拟机调度方案, 具有良好的可并行性. Yan 等人提出一种能耗感知的蚁群优化算法[33]. 他们使用 PABFD 算法生成初始解, 并提出一种创新的蚁群位置转移选择概率的计算方式, 有效减少了集群中活跃服务器的数量. Malekloo 等人将虚拟机放置看作多维度的装箱问题, 给出一种用于虚拟机放置和迁移的多目标蚁群优化算法[34]. 他们结合优化目标改良了蚁群路径选择的启发式因子计算公式和信息素更新规则, 并利用多目标折中的帕雷托最优解选择策略选出较优的虚拟机放置方案. Farahnakian 等人提出一种基于蚁群优化算法的改良 ACS-VMC 方法用于虚拟机整合问题[35]. 他们结合样例中服务器的数量, 使用 *K*-近邻算法获取较优的信息素初始值, 并通过一种兼顾局部收敛和全局搜索的蚁群路径生成方法和信息素更新规则, 以获取功耗和 SLAV 较低的虚拟机放置方案. Ragmani 等人提出一种用于虚拟机调度的混合模糊蚁群优化算法(FACO)[36]. FACO 利用模糊模块对历史信息进行评估, 以计算蚁群路径的信息素值, 降低了算法的运行响应时间和开销. Xiao 等人提出一种基于蚁群系统(ACS)的多目标虚拟机整合方法 DA-VMC[37]. 该文献给出了一种考虑服务器负载程度和虚拟机数量的启发式因子, 结合信息素生成每一只蚂蚁的路径, 在保证 QoS 的同时, 减少了服务器能耗和活跃服务器的数量.

人工蜂群算法模仿蜜蜂搜寻蜜源的行为对虚拟机调度方案进行搜索, 适用于数值优化问题, 并具有较好的鲁棒性. Li 等人提出了一种基于人工蜂群算法的虚拟机整合策略(EC-VMC)[38]. 该策略对服务器的超载欠载概率进行预测, 选出合适的待迁移虚拟机和目标服务器, 并让雇佣蜂、跟随蜂、侦查蜂分别调用 ABC- FVC 算法对虚拟机调度方案进行寻优, 保证调度后每台服务器的资源负载都处于合理高效的水平. Jiang 等人给出了一种基于数据密集型作业的 DataABC 虚拟机调度策略[39]. 他们创新地使用 *k*-means 聚类算法, 根据实时负载情况将服务器集群分为超载、正常、欠载这3类, 并将目标服务器当作食物源, 待迁移虚拟机当作雇佣蜂, 建立放置关系, 通过派遣跟随蜂, 进一步搜索更为节能的虚拟机放置方案.

除了上述 5 种常见的群智能算法外, 萤火虫优化算法(FFO)、灰狼优化算法(GWO)、鲸鱼优化算法(WOA)等多个群智能算法也在虚拟机和任务调度问题上得到了广泛的应用. Yavari 等人考虑服务器的实时功耗和温度两种因素, 提出了一种基于萤火虫优化算法的虚拟机放置算法(FET-VC), 在减少集群能耗开销的同时, 降低了 SLAV[40]. Abdel-Basset 等人利用莱维飞行改进了鲸鱼捕食时的行动模式, 并提出了一种基于莱维飞行的鲸鱼优化算法(ILWOA), 能够高效地找出活跃服务器数量最少的虚拟机调度方案[41]. Al-Moalmi 等人根据二值模型和离散化模型对灰狼优化算法进行改良, 提出了节能效果更优的二值灰狼优化算法(BGWO)和收敛速度更快的离散化灰狼优化算法(DGWO)[42].

通过分析上述的文献, 我们发现主流的群智能虚拟机整合策略大多采用单一的群智能算法, 这些整合策略针对不同的优化目标对算法的流程进行了一定的改进, 能够得到能耗较低的虚拟机调度方案. 然而, 不同的群智能算法存在各自的局限性, 如遗传算法易出现早熟收敛, 粒子群优化算法易陷入局部最优, 人工蜂群算法收敛速度过慢等. 这些限制导致单一群智能算法难以进一步地搜索出更优的虚拟机调度方案. 为此, 本文提出的 HSI-VMC 方法给出了一种结合 DE 和 PSO 的虚拟机放置算法 HDH-DEPSO. 该算法利用 DE 的交叉变异操作对群体进行大幅度扰动, 帮助 PSO 跳出局部最优, 保持群体的多样性; 利用 PSO 的群体最优解和个体最优解引导较优个体在局部范围内搜索出更优的虚拟机放置方案, 加强算法的局部搜索能力.

2 问题定义及优化目标

随着数据中心虚拟机数量的日益增长, 工作负载的频繁变动会导致服务器的超载或欠载, 进而引起虚拟机性能下降和集群能耗升高的问题. 因此, 实现一种高效的虚拟机整合算法是必要的. 本文提出了一种节能高效的 HSI-VMC 方法, 该方法能够结合资源监控器与虚拟机调度器, 共同构成数据中心虚拟机调度系统框架, 如图 1 所示. 在此框架中, 用户向数据中心发出虚拟机创建请求, 数据中心的监视器会定期接收用户的请求, 并收集服务器集群和虚拟机的资源使用信息供 HSI-VMC 方法使用. HSI-VMC 方法会针对集群的负载状

况对虚拟机进行二次调度，并构造出新的“虚拟机-服务器”映射，传递给虚拟机调度器. 虚拟机调度器根据该映射对虚拟机进行迁移，调整集群的负载，达到降低集群功耗和保证服务质量的目的.

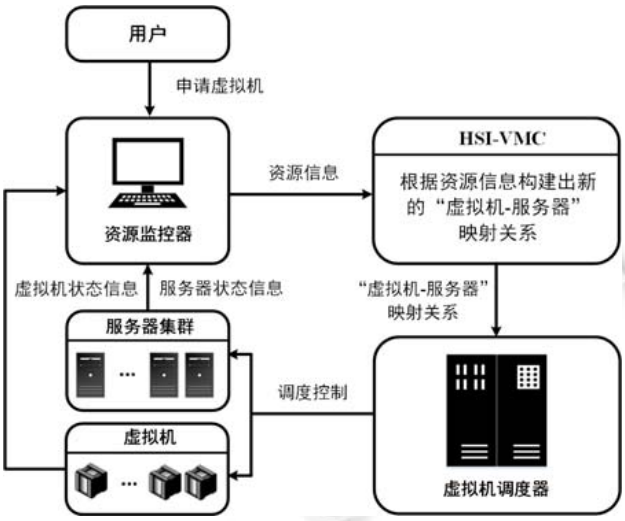


图 1 数据中心的虚拟机调度系统框架

为方便问题的定义,表 1 给出了本文常用的变量及相应的解释.

表 1 变量列表

变量名	描述
H	数据中心中的服务器集合
VMs	数据中心中的虚拟机集合
$x_{i,j}$	记录 vm_j 是否放置在 $host_i$ 上
$P_i(t)$	$host_i$ 在时刻 t 时的功耗
$P_i^{peak\text{-}peff}$	$host_i$ 在峰值效能比下的功耗
$u_{peak\text{-}peff_i}^{CPU}$	$host_i$ 在峰值效能比下的 CPU 利用率
$UP\text{-}THR_i$	$host_i$ 的 CPU 资源上限阈值
$LW\text{-}THR$	当前服务器集群的 CPU 资源下限阈值
$VM_{Migrate}$	待迁移虚拟机列表
$VmList_i$	在 $host_i$ 上运行的虚拟机列表
R	服务器的一种资源(包括 CPU、内存、带宽)
u_i^R	$host_i$ 资源 R 的利用率
R_i^{Cap}	$host_i$ 资源 R 的容量
R_j^{Req}	vm_j 对资源 R 的请求量
D	解向量 SV 的维数，与 R 中的服务器数量相同
$V_{t,G}$	第 G 次迭代中个体 t 的速度向量
V_G	第 G 次迭代中所有个体的速度向量组成的集合
$X_{t,G}$	第 G 次迭代中个体 t 的位置向量，和 SV 意义相同
X_G	第 G 次迭代中所有个体的位置向量组成的集合
S	群智能算法的群体规模
TG	群智能算法的总迭代次数
P	群体中较优个体的占比(应用于 HDH-DEPSO)
MR	差分进化算法中变异操作的概率因子
CR	差分进化算法中交叉操作的概率因子
Y	Discrete-PSO 位置更新中单个解向量需要更新的维数占比

一种高效的“虚拟机-服务器”映射需要兼顾集群功耗和多个 QoS 指标，并尽可能地减少服务器资源碎片. 因此，本文提出的虚拟机整合方法应满足式(1)给出的 3 个目标. 其中，X 表示一种“虚拟机-服务器”映射，该映射通过虚拟机整合方法获得，是集群的虚拟机调度解；F_{Power} 表示数据中心服务器的总功耗；F_{SLAV} 表示云服务

的 SLAV; F_{Active} 表示活跃服务器的数量; SLAV 表示服务水平协议违约率, 用于衡量数据中心的 QoS.

$$\begin{cases}\min F_{power}(X)\\ \min F_{SLAV}(X)\\ \min F_{Active}(X)\end{cases} \tag{1}$$

本文假定数据中心共有 M 台服务器与 N 台虚拟机, 分别用集合 $H=\{host_1,host_2,\ldots,host_M\}$ 和集合 $VMs=\{vm_1, vm_2,\ldots,vm_N\}$ 表示. 虚拟机 vm_j 运行在主机 $host_i$ 上. $R\in\{CPU,RAM,Bw\}$ 表示服务器和虚拟机的资源类型, 不同服务器和虚拟机具有不同的资源容量. 一台虚拟机只能放置到一台服务器中, $x_{i,j}$ 代表虚拟机的放置状态, 如公式(2)所示.

$$x_{i,j}=\begin{cases}1, & \text{if } vm_j\in VmList_i\\ 0, & \text{else}\end{cases},\ \sum_{i=1}^{M}x_{i,j}=1 \tag{2}$$

服务器各种资源的利用率取决于运行在服务器上的虚拟机数量以及资源请求量, 如公式(3)所示, $\sum R_k^{Req}$ 为 $host_i$ 中 vm_k 对资源 R 的请求总量, R_i^{Cap} 表示 $host_i$ 中资源 R 的容量.

$$u_i^R=\frac{\sum R_k^{Req}}{R_i^{Cap}},\ vm_k\in VmList_i \tag{3}$$

虚拟机在进行放置时会受到主机资源的限制, 服务器在每台虚拟机放置后需满足公式(4)的约束.

$$\sum CPU_k^{Req}\leqslant UP\text{-}THR_i,\ \sum RAM_k^{Req}\leqslant RAM_i^{Cap},\ \sum Bw_k^{Req}\leqslant Bw_i^{Cap},\ \forall host_i\in H,\ vm_k\in VmList_i \tag{4}$$

服务器能耗主要来源于 CPU、内存、磁盘、网卡等部件, CPU 的能耗开销更是其中的主体. 一台服务器的实时功耗与自身的 CPU 利用率呈近似线性关系, 利用 SPECpower 提供的服务器不同负载程度下的能耗数据, 可以计算出服务器的实时功率. 单个服务器的实时功率 $P_i(t)$ 的计算如公式(5)所示, $u_i(t)$ 为 $host_i$ 在 t 时刻的 CPU 利用率. 对 SPECpower 的服务器数据进行线性插值, 可得到与时间 t 相关的分段函数:

$$P_i(t)=\begin{cases}k_1u_i(t)+b_1, & 0\leqslant u_i(t)<0.1\\ k_2u_i(t)+b_2, & 0.1\leqslant u_i(t)<0.2\\ k_3u_i(t)+b_3, & 0.2\leqslant u_i(t)<0.3\\ \quad\vdots\\ k_{10}u_i(t)+b_{10}, & 0.9\leqslant u_i(t)\leqslant 1\end{cases} \tag{5}$$

对 $P_i(t)$ 进行积分, 可得 $host_i$ 在 t_1–t_2 时间段所产生的能耗, 如公式(6)所示, 数据中心中所有服务器的能耗如公式(7)所示.

$$E_i=\int_{t_1}^{t_2}P_i(t)\mathrm{d}t \tag{6}$$

$$E=\sum_{i=1}^{m}E_i \tag{7}$$

通过研究 SPECpower 中 20 余款不同类型服务器的能效数据, 我们可以得到计算效能比的近似函数, 如公式(8)所示.

$$peff\approx\frac{ps_i(u)}{P_i(u)} \tag{8}$$

其中, u 为服务器的 CPU 利用率, $ps_i(u)$ 和 $P_i(u)$ 分别为 $host_i$ 在 CPU 利用率为 u 时的性能与能耗. 其中, 性能的衡量指标为服务器每秒处理的指令数. 在性能需求一定的条件下, 效能比越高的服务器功率越低. 因此, 虚拟机调度器应该尽可能将虚拟机放置到效能比较高的服务器上, 并使各服务器维持在峰值效能比状态下运行.

3 混合群智能节能虚拟机调度方法设计

3.1 差分进化算法与粒子群优化算法

差分进化算法通过变异、交叉、选择这 3 种操作的多次迭代对群体进行更新, 以获取较优的解. 个体的

变异操作如公式(9)所示. $X_{r_1,G}$, $X_{r_2,G}$, $X_{r_3,G}$ 为随机选出的 3 个个体, F 为放缩常量 $M_{t,G}$ 为变异个体.

$$M_{t,G} = X_{r_1,G} + F \cdot (X_{r_2,G} - X_{r_3,G}) \tag{9}$$

交叉操作以位运算的方式确定交叉个体 $C_{t,G}$, 如公式(10)所示. $M_{t,t,G}$, $X_{t,t,G}$, $C_{t,t,G}$ 分别为群体的第 t 个交叉个体、原始个体、交叉个体第 i 位的值, CR 为交叉概率, $rand(i)$为大于 0 小于 1 的随机实数.

$$C_{i,t,G} = \begin{cases} X_{i,t,G}, & rand(i) \leqslant CR \\ M_{i,t,G}, & rand(i) > CR \end{cases} \tag{10}$$

在一次迭代中, 算法执行 S 次变异和交叉操作, 得到变异群体 M_G 和交叉群体 C_G. 最后, 通过选择操作, 从原始群体 X_G 和交叉群体 C_G 中分别取出对应的两条解向量, 并通过贪心选择策略, 保留更优的解向量作为新一代的个体.

粒子群优化算法将个体视作问题的解, 在每次迭代中, 依次执行速度更新、位置更新、个体最优解更新、群体最优解更新这 4 个步骤对群体进行优化, 得到较优的解. 个体的速度更新操作如公式(11)所示. 第 t 个体更新后的速度 $V_{t,G}$ 受到上一次迭代的速度 $V_{t,G-1}$、个体最优解 $X_{t,G-1}^{ibest}$、群体最优解 X_{G-1}^{gbest} 以及个体位置 $X_{t,G-1}$ 的影响. 其中, w 表示个体的惯性因子, c_1 和 c_2 为加速常数, r_1 和 r_2 为随机常数.

$$V_{t,G} = wV_{t,G-1} + c_1 r_1 (X_{t,G-1}^{ibest} - X_{t,G-1}) + c_2 r_2 (X_{G-1}^{gbest} - X_{t,G-1}) \tag{11}$$

个体的位置更新操作如公式(12)所示, 其本质是利用本轮迭代中获取的速度 $V_{t,G}$ 更新自身在上一轮迭代中的位置 $X_{t,G-1}$, 得到新的位置向量 $X_{t,G}$.

$$X_{t,G}=X_{t,G-1}+V_{t,G} \tag{12}$$

在每次迭代中, 算法都会根据公式(11)、(12)计算出每一个体的速度、位置及相应的适应度函数值, 并对个体最优解 $X_{t,G-1}^{ibest}$ 和全局最优解 X_G^{gbest} 进行更新.

3.2 混合群智能节能虚拟机整合方法(HSI-VMC)的设计

本文提出的 HSI-VMC 方法对超载和欠载服务器进行检测, 并选出合适的虚拟机进行迁移, 以达到虚拟机整合的目的. 如图 2 所示, HSI-VMC 方法主要分为 4 个阶段: (1) 检测超载服务器, 并从中选出待迁移虚拟机; (2) 选出合适的目标服务器, 作为待迁移虚拟机的迁移目标; (3) 完成待迁移虚拟机到目标服务器的放置; (4) 根据虚拟机迁移后的集群负载情况选出欠载服务器, 并重新放置其中的虚拟机, 关断空闲服务器. 其中, 阶段 3 和阶段 4 使用本文提出的 HDH-DEPSO 算法进行虚拟机放置, 以获取高效合理的“虚拟机-服务器”映射. 下面我们将对 HSI-VMC 方法的各个阶段进行详细的介绍.

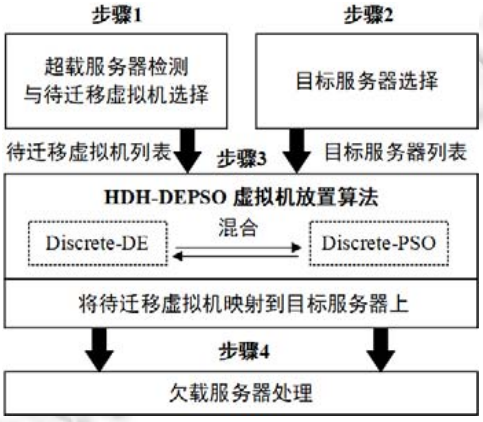


图 2 HSI-VMC 方法流程图

3.2.1 超载服务器检测与待迁移虚拟机选择策略

在第 1 阶段, 我们利用一种基于峰值效能比的静态阈值超载服务器检测策略对超载服务器进行检测, 并

根据迁移价值比选出合适的虚拟机从超载服务器中迁出. 通过查阅 SPECpower 的服务器实时能效数据, 我们可以得到不同型号服务器峰值效能比下的 CPU 利用率 $u_{peak\text{-}peff}^{CPU}$. 我们发现: 大部分服务器的 $u_{peak\text{-}peff}^{CPU}$ 集中在 60%−90%之间, 部分服务器的 $u_{peak\text{-}peff}^{CPU}$ 为 100%; 而在 $u_{peak\text{-}peff}^{CPU}$ ±10% 的波动范围内, 服务器的效能比与自身的峰值效能比 $peak\text{-}peff_i$ 相近. 为了充分利用服务器的资源并避免服务器超载的频繁发生, 本文根据 $u_{peak\text{-}peff}^{CPU}$ 为不同类型的服务器设定了超载触发阈值, 如公式(13)所示.

$$UP\text{-}THR_i = \min(u_{peak\text{-}peff}^{CPU_i} + 10\%, 90\%) \tag{13}$$

CPU 利用率大于 $UP\text{-}THR_i$ 的服务器会加入超载服务器列表 $H_{Overload}$ 中, 我们需要选择合适的虚拟机迁出以降低超载服务器的负载程度. 为了保证 QoS, 降低虚拟机迁移所带来的性能下降幅度, 我们采用了文献[43]中 Lin 等人提出的基于迁移价值比的待迁移虚拟机选择策略. 该策略会根据每个虚拟机的迁移价值比 *Migra-Ratio* 选出合适的虚拟机进行迁移, 以减少待迁移虚拟机的数量. *Migra-Ratio* 的计算公式如式(14)所示.

$$Migra\text{-}Ratio_k = |CPU_i^{Overload} - CPU_k^{Req}| \cdot \frac{RAM_k^{Req}}{Bw_i^{avail}},\ \forall host_i \in H,\ vm_k \in VmList_i \tag{14}$$

$CPU_i^{Overload}$ 表示 $host_i$ 的实时负载超出 $UP\text{-}THR_i$ 的 CPU 资源量, CPU_k^{Req} 表示虚拟机的 CPU 资源请求量, RAM_k^{Req} 表示虚拟机的内存资源请求量, Bw_i^{avail} 表示 $host_i$ 可用于迁移的带宽. 每次选择 *Migra-Ratio* 最小的虚拟机迁出, 直至该服务器的 CPU 负载小于 $UP\text{-}THR_i$ 为止.

被选出的虚拟机共同组成待迁移虚拟机列表 $VM_{Migrate}$.

3.2.2 超载服务器检测与待迁移虚拟机选择策略

在算法的第 2 阶段, 我们从活跃服务器集群中选出部分服务器构成目标服务器列表 H_{Target} 用于虚拟机的再分配. 目标服务器选择的约束条件如公式(15)所示.

$$\forall host_i \in H_{Active},\ \forall host_i \notin H_{Overload},\ \exists vm_k \in VM_{Migrate} \text{ s.t. } host_i \text{ satisfies formula} \tag{15}$$

其中, H_{Active} 为活跃服务器列表, $VM_{Migrate}$ 为待迁移虚拟机列表. 公式(15)表示目标服务器必须为活跃服务器且不在超载服务器列表 $H_{Overload}$ 中; 同时, 该服务器至少能够成功放置一台待迁移虚拟机.

3.2.3 虚拟机放置策略

在选定 $VM_{Migrate}$ 列表和 H_{Target} 列表后, 整合方法进入虚拟机放置阶段. 我们采用混合离散化启发式差分进化粒子群优化算法搜索节能高效的虚拟机放置方案. 在该算法中, 每个个体均表示一种虚拟机放置方案. 个体的每一维度均代表目标服务器 $host_i$ 上运行的待迁移虚拟机集合. 由于每轮虚拟机调度仅负责待迁移虚拟机的放置, 因此个体仅表示本次算法流程中待迁移虚拟机的分配状况, 解向量中某一维 $host_i$ 的虚拟机集合可能为空, 如图 3 所示.

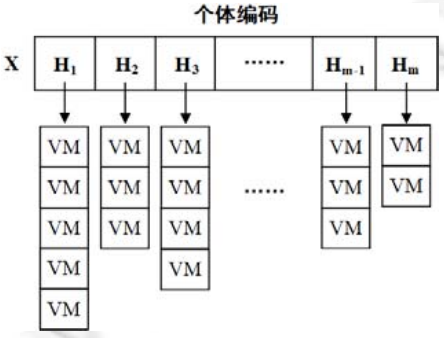


图 3 个体(解向量)编码方式

适应度函数用于评价个体的优劣程度. 我们给出了一种基于加权欧氏距离的适应度函数, 使得各服务器的 CPU 利用率能够尽可能保持在 $u_{peak\text{-}peff}^{CPU}$ 附近, 并减少服务器集群的总功耗, 如公式(16)所示. 其中, $P_{\max_i}$ 为目标服务器 $host_i$ 的峰值功耗, $P_{tar\text{-}\max}$ 为 H_{Target} 中最大的服务器峰值功耗, w_1, w_2 分别为两种影响因子的权重.

适应度函数值越小，相应的虚拟机放置方案越优.

$$f(X_i)=\sum_{i=1}^{D}\sqrt{w_1(u_i^{CPU}-u_{peak\text{-}peff_i}^{CPU})^2+w_2\left(\frac{P_i}{P_{tar\text{-}max}}\right)^2},\ w_1+w_2=1,\ P_{tar\text{-}max}=\max(P_{\max_i},...,P_{\max_D}) \tag{16}$$

HDH-DEPSO 在进入迭代前，需要生成初始群体. 为了保证群体的质量和多样性，本文采用了 PEAP[43], MBFD[12], RANDOM-FFD 这 3 种启发式虚拟机放置算法. 利用 PEAP, MBFD 生成 2 个解，剩余的 n–2 个解则通过 RANDOM-FFD 生成. PEAP 将虚拟机放置到剩余最优 CPU 资源量最符合自身资源需求的服务器上. 若由于 RAM 和 Bandwidth 资源不足而导致虚拟机放置失败，则优先将虚拟机放置到所有目标服务器中峰值效能比最高的服务器上. 该算法能够有效地减少资源碎片，使服务器保持在峰值效能比下工作. MBFD 将 $VM_{Migrate}$ 列表的虚拟机按 CPU 资源请求量降序排序，然后选择能耗增量最小的服务器进行虚拟机放置. RANDOM-FFD 随机选择服务器进行虚拟机放置，若虚拟机在给定的次数内放置失败，则利用 FFD 对虚拟机进行放置. 若初始群体生成过程中出现资源不足的情况，则需要执行服务器开启策略. 本文提出了一种基于欠载资源量的贪心服务器开启策略，尽可能少地开启服务器，进而减少服务器集群的总能耗. 该策略通过统计 $VM_{Migrate}$ 列表中未被放置虚拟机不同资源的请求总量 $\sum CPU_k^{Req}$, $\sum RAM_k^{Req}$, $\sum Bw_k^{Req}$，优先选出满足公式(17)且 $UP\text{-}THR$ 最小的一台服务器开启. 若不存在满足条件的服务器，则开启 $UP\text{-}THR_i$ 最接近 $\sum CPU_k^{Req}$ 的服务器. 最后，将开启的服务器加入目标服务器列表 H_{Target} 中，并更新群体的信息.

$$UP\text{-}THR_i \geqslant \sum CPU_k^{Req},\ RAM_i^{Cap} \geqslant \sum RAM_k^{Req},\ Bw_i^{Cap} \geqslant \sum Bw_k^{Req},\ \forall host_i \in H,\ \forall vm_k \in VM_{Reallocate} \tag{17}$$

在初始解生成后，HDH-DEPSO 放置算法正式进入迭代阶段，其迭代过程如图 4 所示. 首先，个体 X_i 按优劣程度降序排序组成群体；然后，原始群体通过差分进化算法的变异和交叉操作对所有个体进行大幅度随机扰动，构造出变异群体和交叉群体；差分进化算法的选择操作会根据适应度函数值，使用轮盘赌算法从原始群体中较差的 $S\cdot(1–P\%)$的个体、变异群体、交叉群体中选出 $S\cdot(1–P\%)$的个体作为新群体的一部分；最后，利用粒子群优化算法对群体中较优的 $S\cdot P\%$的个体进行优化，生成新群体剩余的 $S\cdot P\%$的个体. HDH-DEPSO 利用差分进化算法的交叉和变异操作对个体进行随机扰动，帮助粒子群优化算法跳出局部最优，利用粒子群优化算法的群体最优解和个体最优解为搜索过程提供寻优方向，加强了算法的局部收敛能力.

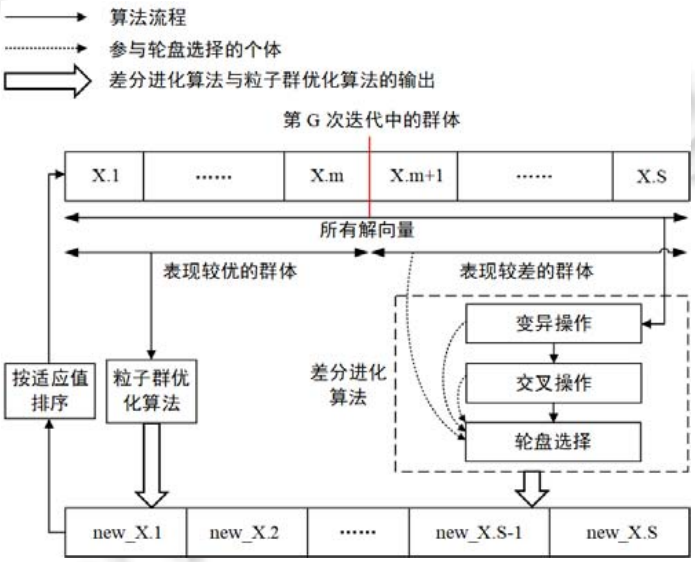


图 4 HDH-DEPSO 算法的迭代过程

(一) 离散化差分进化算法

原始的 DE 算法操作采用连续型实数向量，并不适用于解决虚拟机调度问题. 因此，本文提出了一种离散

化差分进化算法(discrete-DE), 针对集合向量重定义了差分进化算法的 3 个步骤. 在变异操作方面, 我们分别对“–”和“+”运算进行了重定义, 得到了“⊙”和“⊕”运算. 将“⊙”运算重新定义为一种更新解向量的特殊运算, F 重定义为概率值 MR, 作为“⊙”运算的参数. 将“⊕”运算重新定义为处理虚拟机放置冲突的启发式算法. 变异操作见公式(18).

$$M_{t,G}=X_G^{Remain}\oplus MR\cdot(X_{t,G}\odot X_{r_2,G}),\ (t=1,2,...,S) \tag{18}$$

如图 5 所示, 在 $X_{t,G}$ 和 $X_{r_2,G}$ 第 i 维的“⊙”运算中, $H_{t,i}^{VM}$ 和 $H_{r_2,i}^{VM}$ 表示个体 t 和个体 r_2 第 i 维服务器的虚拟机集合. MR 为虚拟机保留在第 i 维服务器上的概率. “⊙”运算的具体步骤为: (1) 对两个虚拟机集合分别作交运算和 $X_{t,G}$ 对 $X_{r_2,G}$ 的差运算, 得到虚拟机集合的交集 DF_i''' 和差集 DF_i; (2) DF_i 中的虚拟机有 MR 的概率加入虚拟机集合 DF_i' 中, 有 $1–MR$ 的概率加入虚拟机集合 DF_i'' 中, 其中, DF_i' 为需要重新分配的虚拟机集合 $VM_i^{reallocate}$; (3) DF_i'' 与 DF_i''' 作并运算, 得到保留在 $host_i$ 上的虚拟机集合 $X_{i,G}^{Remain}$; (4) 对 $X_{t,G}$ 和 $X_{r_2,G}$ 的每一维执行步骤(1)–步骤(3)后, “⊙”运算完成.

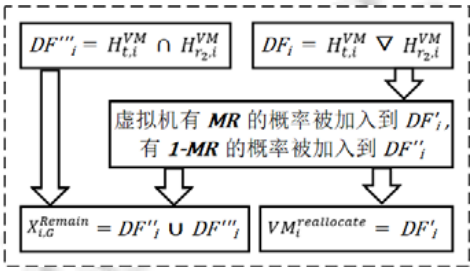


图 5 $X_{t,G}$ 和 $X_{r_2,G}$ 第 i 维的“⊙”运算

为了更好地解释“⊙”运算前 3 步的操作步骤, 这里将给出一个例子. 假设 MR=100%, 解向量 $X_{t,G}$ 中第 i 维服务器的虚拟机集合为 $H_{t,i}^{VM}=\{10,1,8,7\}$, 而解向量 $X_{r_2,G}$ 中第 i 维服务器的虚拟机集合为 $H_{r_2,i}^{VM}=\{2,3,8,9\}$, 则两者第 i 维服务器的虚拟机集合作“⊙”运算得到的 $VM_i^{reallocate}$ 为{1,2,3,7,9,10}, $X_{i,G}^{Remain}$ 为{8}. 在 $X_{t,G}$ 和 $X_{r_2,G}$ 的各维度均完成“⊙”运算后, 我们会得到解向量不同维度的 $VM_i^{reallocate}$ 和 $X_{i,G}^{Remain}$. 如公式(19)所示, 对 $VM_i^{reallocate}$ 取并集可以得到需要重新分配的虚拟机集合 $VM_i^{reallocate}$, $X_{i,G}^{Remain}$ 将组合成公式(18)的中 X_G^{Remain}.

$$VM^{reallocate}=VM_1^{reallocate}\cup VM_2^{reallocate}\cup...\cup VM_D^{reallocate},\ X_G^{Remain}=(X_{1,G}^{Remain},...,X_{D,G}^{Remain}) \tag{19}$$

接下来, 我们需要将集合 $VM^{reallocate}$ 中的虚拟机重新放置到 X_G^{Remain} 的各维服务器当中, 这一操作定义为公式(18)中的“⊕”运算. 我们会使用改良的最佳匹配递减算法(MBFD)将 $VM^{reallocate}$ 的虚拟机重新放置到服务器中. 在一次算法迭代中, 我们需要完成 S 次公式(18)的变异操作, 并生成 S 个变异向量 $M_{t,G}$. 所有变异向量 $M_{t,G}$ 共同组成本次迭代的变异群体 M_G.

M_G 构造完成后, 算法对每个原始个体 $X_{t,G}$ 和变异个体 $M_{t,G}$ 执行公式(20)的“×”运算, 生成个体 $XM_{t,G}$, 并把冲突的虚拟机重新分配到个体 $XM_{t,G}$ 中, 得到交叉个体 $C_{t,G}$.

$$XM_{t,G}=X_{t,G}\times M_{t,G} \tag{20}$$

交叉操作的详细流程如图 6 所示, 其中, H_i 为解向量第 i 维的服务器. 具体步骤如下: (1) 取出个体 $X_{t,G}$, $M_{t,G}$; (2) 以交叉概率 CR 决定应该取 $X_{t,G}$ 还是 $M_{t,G}$ 的第 i 维虚拟机集合作为 $XM_{t,G}$ 第 i 维的值, 对 $X_{t,G}$ 和 $M_{t,G}$ 的每一维执行上述操作生成 $XM_{t,G}$; (3) 将“×”运算中放置冲突的虚拟机从服务器中取出, 并与未得到放置的虚拟机一起加入集合 $VM^{unallocate}$ 中; (4) 使用 MBFD 算法将 $VM^{unallocate}$ 中的虚拟机重新放置到 $XM_{t,G}$ 的服务器中, 得到交叉个体 $C_{t,G}$, 交叉操作结束. 第 G 次迭代群体中的所有个体 $X_{t,G}$ 与变异群体中的个体 $M_{t,G}$ 执行交叉操作后, 生成 S 个交叉个体 $C_{t,G}$, 共同构成交叉群体 C_G.

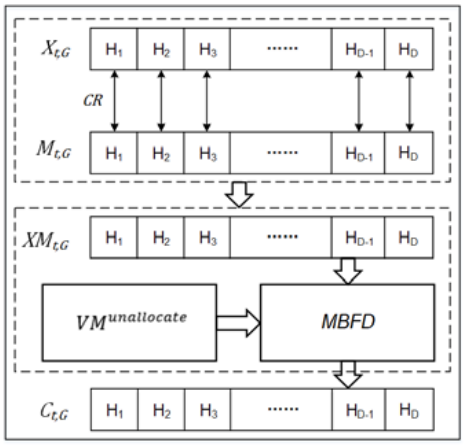


图 6 $X_{t,G}$ 和 $M_{t,G}$ 的交叉操作流程

在执行变异和交叉操作后，我们会得到变异群体 M_G 和交叉群体 C_G. 这两个群体与第 G 次迭代的群体中适应度函数值较高的 1–P%的个体共同参与 Discrete-DE 的选择操作. 对上述 3 个群体中共 S·(3–P%)的个体进行排序，并在较优的一半中采用轮盘赌的方式选出 S·(1–P%)的个体作为新群体的一部分. 轮盘赌中，个体被选中的概率由自身的适应度函数值决定，适应度值较小的解会有更大的概率被选中. 至此，Discrete-DE 执行完毕.

(二) 离散化粒子群优化算法

如公式(21)、(22)所示，本文重新定义了 PSO 的速度更新和位置更新操作.

$$V_{t,G} = P_1 V_{t,G-1} + P_2 X_{t,G-1}^{ibest} + P_3 X_{G-1}^{gbest} \tag{21}$$

$$X_{t,G} = X_{t,G-1} + V_{t,G} \tag{22}$$

其中，$V_{t,G\text{-}1}$，$X_{t,G-1}^{ibest}$，X_{G-1}^{gbest} 分别为第 G–1 次算法迭代的速度向量、个体最优解、群体最优解. PSO 的速度更新操作利用速度惯性、全局最优解、局部最优解生成新的速度向量. 本文以概率的形式重新定义了个体的速度更新操作，其中，P_1, P_2, P_3 为概率值，其计算方式如公式(23)–(25)所示.

$$P_1 = \frac{1/f(V_{t,G-1})}{1/f(V_{t,G-1}) + 1/f(X_{t,G-1}^{ibest}) + 1/f(X_{G-1}^{gbest})} \tag{23}$$

$$P_2 = \frac{1/f(X_{t,G-1}^{ibest})}{1/f(V_{t,G-1}) + 1/f(X_{t,G-1}^{ibest}) + 1/f(X_{G-1}^{gbest})} \tag{24}$$

$$P_3 = \frac{1/f(X_{G-1}^{gbest})}{1/f(V_{t,G-1}) + 1/f(X_{t,G-1}^{ibest}) + 1/f(X_{G-1}^{gbest})} \tag{25}$$

第 G 次迭代中，更新得到的速度 $V_{t,G}$ 表示一种虚拟机放置方案. 公式(21)表示 $V_{t,G-1}$，$X_{t,G-1}^{ibest}$，X_{G-1}^{gbest} 第 i 维虚拟机集合中的虚拟机分别有 P_1, P_2, P_3 的概率保留在 $V_{t,G}$ 第 i 维的服务器中. 若同一虚拟机同时存在于上述 3 个虚拟机集合的多个集合中，则其保留概率叠加. 我们根据保留概率选取虚拟机，并逐一放置到 $V_{t,G}$ 第 i 维的服务器中. 图 7 给出了 $V_{t,G-1}$ 中第 1 维更新的一个例子，其中，$V_{t,G}$:$H1^{VM}$，$X_{t,G-1}^{ibest}$:$H1^{VM}$，X_{G-1}^{gbest}:$H1^{VM}$，$V_{t,G}$:$H1^{VM}$ 分别为 $V_{t,G-1}$，$X_{t,G-1}^{ibest}$，X_{G-1}^{gbest}，$V_{t,G}$ 第 1 维服务器的虚拟机集合. VM_1 有 P_1+P_2 的概率加入 $V_{t,G}$:$H1^{VM}$ 中，VM_7, VM_9 有 $V_{t,G}$:$H1^{VM}$ 的概率加入 $V_{t,G}$:$H1^{VM}$ 中，VM_5, VM_{10} 有 P_1 的概率加入 $V_{t,G}$:$H1^{VM}$ 中，VM_6 有 P_2 的概率加入 $V_{t,G}$:$H1^{VM}$ 中，VM_2, VM_4 有 P_3 的概率加入 $V_{t,G}$:$H1^{VM}$ 中. 需要注意：在速度更新操作中，服务器可能会出现虚拟机数量过多、资源不足的情况，我们采用一种尽力而为的方法，将更多的虚拟机放置到同一维的服务器中.对于没有得到放置的虚拟机，我们会利用 MBFD 将它们放置到 $V_{t,G}$ 的各维服务器中. 对所有个体的速度 $V_{t,G-1}$ 执行更新操作后，本轮算法的速度更新操作执行完毕.

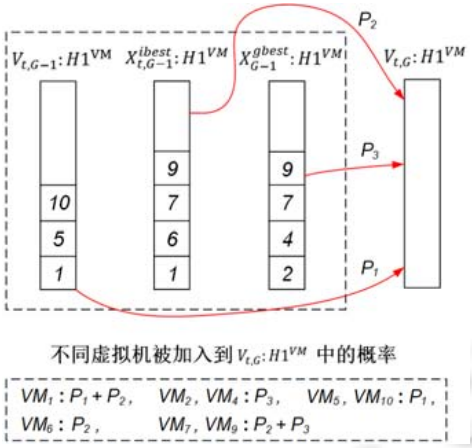


图 7 个体速度单一维度的更新实例

在位置更新操作中，我们利用本轮迭代得到的速度 $V_{t,G}$ 对群体中较优的 P%的个体的位置向量 $X_{t,G-1}$ 进行更新. 为了保证较优个体的稳定性，公式(22)的位置更新操作仅对每个个体中 Y%维服务器的虚拟机集合进行更新. 从 $X_{t,G-1}$ 中随机选出 Y%的服务器，并将这部分服务器的虚拟机集合替换为速度向量 $V_{t,G}$ 相应维数上的虚拟机集合，得到 $X_{t,G}^{incomplete}$. $X_{t,G}^{incomplete}$ 中可能存在未得到放置的虚拟机，我们使用 MBFD 算法将这部分虚拟机放置到 $X_{t,G}^{incomplete}$ 各维的服务器中，得到更新后的位置向量 $X_{t,G}$，完成位置更新操作. 速度更新和位置更新操作完成后，算法会根据个体的适应度函数值对个体最优解和群体最优解进行更新. 至此，Discrete-PSO 的一次迭代执行完毕.

Discrete-DE 和 Discrete-PSO 共同组成 HDH-DEPSO. 在 HDH-DEPSO 进行多次迭代后，我们会选取全局最优解向量作为最终输出，并构建出需要进行迁移的虚拟机序列. HDH-DEPSO 虚拟机放置算法的伪代码如图 8 所示.

Algorithm 1. HDH-DEPSO virtual machine placement.
Input: H_{Target}, $VM_{Migrate}$;
Output: *MigrationMap*.

```
1:
2:   Initialize the Swarm
3:
4:   for G from 1 to TG do
5:     Use Eq.(16) to Update f(X_{t,G}) of X_{t,G}
6:     Sort Swarm by f(X_i) in ascending order
7:
8:     X_G←Mutation(Swarm)
9:     C_G←Crossover(M_G,Swarm)
10:    Update X^{ibest} and X^{gbest} by M_G and C_G
11:
12:    NewSwarm←Select(Swarm_{worsepart},M_G,C_G)
13:
14:    V_G←VelocityUpdate(Swarm_{betterpart})
15:    NewSwarm, X_G←PositionUpdate(Swarm_{betterpart})
16:
17:    Update X^{ibest} and X^{gbest} by NewSwarm
18:    Update X^{gbest} by V_G
19:  end for
20:
21:  Construct the MigrationMap and return
```

图 8 HDH-DEPSO 虚拟机放置算法

3.2.4 欠载服务器处理策略

经过第 3.2.1 节–3.2.3 节的算法流程，$VM_{Migrate}$ 中的待迁移虚拟机全部放置完毕. 为了进一步减少活跃服务

器的数量, 本文提出了一种基于负载均值的欠载服务器处理策略(AVG), 图 9 所示. 首先, 我们选出 CPU 利用率低于欠载触发阈值(*LW-THR*)的服务器组成待处理服务器列表(H_{Wait}), *LW-THR* 为活跃服务器的 CPU 利用率平均值; 然后, H_{Wait} 中 CPU 负载较高的一半服务器和不在 $H_{Overload}$ 与 H_{Wait} 中的服务器共同组成目标服务器列表(H_{Target}), H_{Wait} 中 CPU 负载较低的一半服务器加入欠载服务器列表 $H_{Underload}$ 中; 最后, 将所有欠载服务器上的虚拟机加入 $VM_{Reallocate}$ 列表中, 并关断欠载服务器, 再利用 HDH-DEPSO 将 $VM_{Reallocate}$ 列表中所有的虚拟机重新放置到 H_{Target} 的各台服务器中. 至此, HSI-VMC 方法执行完毕, 算法结束.

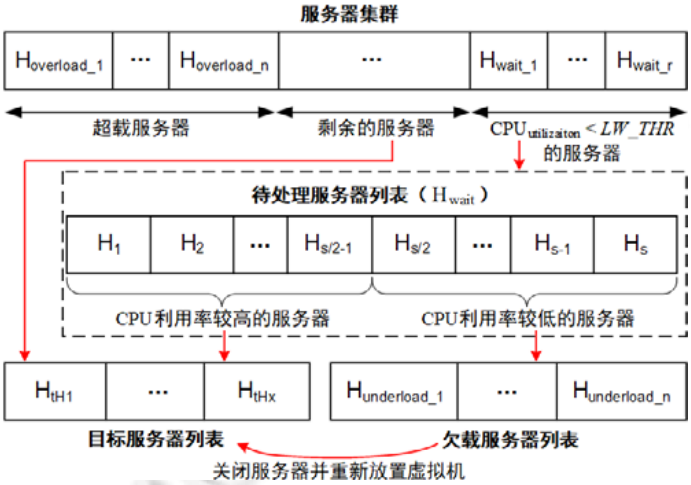


图 9 欠载服务器处理策略

3.3 时间复杂度

HSI-VMC 方法执行一次调度的时间开销由 3 部分组成.

(1) 在超载服务器检测和待迁移虚拟机选择阶段中, 其时间开销主要来源于每台超载服务器上的虚拟机排序. 假设每台超载服务器中的虚拟机数量为 X_i, 在最差情况下, 对每台超载服务器上的虚拟机分别进行排序的时间复杂度为 $O\left(\sum X_i^2\right)$.

(2) 虚拟机放置阶段的时间复杂度取决于群智能算法的操作. HDH-DEPSO 共需执行 $TG{\cdot}S$ 次公式(18)、(20)–(22)的变异、交叉、速度更新、位置更新操作. 假设待放置虚拟机的数量为 P, 目标服务器的数量为 Q, 放置发生冲突的虚拟机数量为 R, 在这 4 种操作中, 每种操作都需要对每台待放置虚拟机进行常数次的访问, 且当放置冲突的虚拟机要执行 MBFD 算法进行重新放置(MBFD 的时间复杂度为 $O(Q{\cdot}R)$, 故 HDH-DEPSO 的时间复杂度为 $O(TG{\cdot}S{\cdot}(P+Q{\cdot}R))$.

(3) 欠载服务器处理部分, 假设集群中活跃服务器的数量为 M. 我们要计算集群的负载均值, 并按照 CPU 利用率对 Host 进行排序, 在最差情况下, 时间复杂度为 $O(M^2)$. 由于我们需要利用 HDH-DEPSO 重新放置需要关断的服务器中的虚拟机, 因此欠载服务器处理阶段时间复杂度为 $O(M^2+TG{\cdot}S{\cdot}(P+Q{\cdot}R))$.

4 实验结果

4.1 实验环境与评价指标

本文以仿真实验的方式, 对 HSI-VMC 的表现进行评估, 在 CloudSim 4.0 平台上设计并执行了相关实验. CloudSim 平台搭建在一台计算机上, 计算机的参数为: Intel(R) Core(TM) i7-8550U CPU@1.80 GHz 1.99 GHz, RAM 为 16.00 GB. 为了构建异构云数据中心环境, 我们总共选取了 6 种型号的服务器、8 种虚拟机实例类型进行实验. 如表 2 和表 3 所示, 服务器的能耗和峰值性能参数取自 SPECpower[43], 虚拟机实例类型的各项参

数参考 Amazon EC2 获得. 图 10 展示了仿真实验所选的 6 款服务器随负载变化的能耗曲线.

表 2 服务器类型

Model	Cores	MIPs/core	RAM (MB)	Bw (Gb/s)	Storage (GB)	*peak-peff*	$u^{peak\text{-}peff}$
Dell PowerEdge R630	36	2 300	65 536	10	1 000	11 623	0.8
Dell PowerEdge R640	56	2 500	196 608	10	1 000	13 420	0.7
Huawei Fusion Server 5288 V5	56	2 700	196 608	10	1 000	17 285	0.8
Acer AR380 F2	12	2 500	32 768	10	1 000	3 904	1.0
Acer AT350 F2	16	2 400	131 072	10	1 000	3 352	0.8
IBM System x3530 M4	16	2 300	24 576	10	1 000	5 942	0.7

表 3 虚拟机实例类型

Type	Cores	MIPs	MEM (MB)	BW (Mb/s)	Storage (MB)
a1.m	1	2 300	2 048	100	3 000
a1.l	2	4 600	4 096	100	3 000
a1.xl	4	9 200	8 192	100	3 000
a1.2xl	8	18 400	16 384	100	3 000
a1.4xl	16	36 800	32 768	100	3 000
m5a.l	2	5 000	8 192	100	3 000
g4dn.xl	4	10 000	16 384	100	3 000
g4dn.2xl	8	20 000	32 768	100	3 000

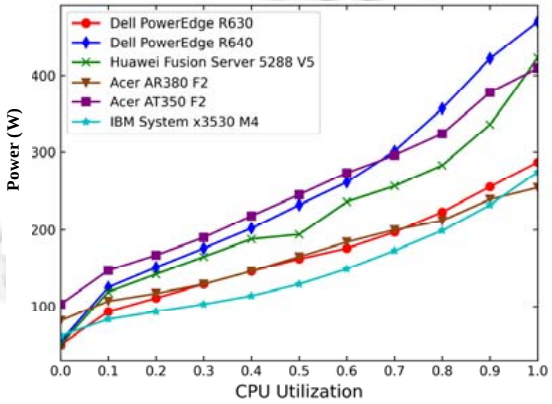


图 10 服务器功耗曲线图

为测试虚拟机整合算法在不同规模集群下的表现, 我们设置了 5 种规模的集群进行实验, 见表 4. 虚拟机负载数据集取自实际的工作负载数据集 PlanetLab 和基于 PlanetLab 数据集生成的 MIX 以及 Gan 数据集. PlanetLab 数据集采集了 2011 年 3 月和 4 月中 10 天的 Cloudlet 负载数据. 我们按照设定的集群规模的大小, 从 PlanetLab 各天的负载数据中抽取出一部分, 作为 Small, Medium, Large 这 3 种规模集群的负载数据. 特别地, PlanetLab-20110306 的负载数据中只有 898 个 Cloudlet 的负载记录, 我们随机选择了当天的 102 个 Cloudlet 的记录进行复制, 并添加到这一天的负载数据中, 以构造出 Large 规模的负载数据. 由于 PlanetLab 各天的负载规模较小, 不足以测试虚拟机整合方法在虚拟机数目较为庞大时的表现, 我们基于 PlanetLab 的数据, 分别采用随机抽样和对抗神经网络生成了 X-Large 和 XX-Large 这 2 种规模的负载数据集 MIX 和 Gan. 3 个数据集的负载属性分别见表 5–表 7. 在我们的实验中, 每个 Cloudlet 对应集群中的一台虚拟机, Cloudlet 的大小服从泊松分布[43]. 在虚拟机的选型上, 我们以轮询分配的方式将 Cloudlet 放置到 8 种类型的虚拟机上进行实验, 使得各类型的虚拟机数量尽可能地相近. 无论是 PlanetLab 还是 MIX 和 Gan 数据集, 三者均记录了不同 Cloudlet 单日的负载状况, 每个 Cloudlet 以 5 分钟为间隔记录 CPU 负载(共 288 条). 实验中, HDH-DEPSO 的算法参数设置见表 8, w_1, w_2 为适应度函数两种影响因子的权重.

表 4 集群规模设定

Scale	Small	Medium	Large	X-Large	XX-Large
Number of Hosts	30	100	300	600	1 200
Number of VMs	160	560	1 000	2 000	4 000

表 5 PlanetLab 负载数据属性表

Date	Number of *VMs*=160		Number of *VMs*=560		Number of *VMs*=1000	
	Mean (%)	*StDev* (%)	*Mean* (%)	*StDev* (%)	*Mean* (%)	*StDev* (%)
03/03/2011	**12.11**	15.81	**11.81**	15.76	**12.06**	16.69
03/06/2011	11.06	15.45	11.14	16.07	11.66	17.19
09/03/2011	10.82	15.51	**11.19**	15.79	10.72	15.58
22/03/2011	9.95	13.12	**9.56**	13.23	**9.23**	12.94
25/03/2011	11.68	16.65	11.13	14.84	10.57	14.13
03/04/2011	11.54	14.45	11.48	15.29	11.93	15.89
09/04/2011	**11.41**	15.02	10.50	13.75	10.92	14.79
11/04/2011	10.70	13.19	11.20	14.21	11.56	15.00
12/04/2011	11.43	14.64	11.39	14.96	**11.45**	15.10
20/04/2011	**9.81**	13.41	10.27	14.79	10.53	15.30

表 6 Mix 负载数据属性表

Scale	Number of *VMs*=2000									
	Mix-1	Mix-2	Mix-3	Mix-4	Mix-5	Mix-6	Mix-7	Mix-8	Mix-9	Mix-10
Mean (%)	10.69	10.72	11.35	**11.43**	**11.03**	10.44	11.17	11.20	11.26	**10.30**
StDev (%)	14.58	14.82	15.66	**15.93**	**14.91**	**14.32**	15.77	15.20	15.16	14.41

表 7 Gan 负载数据属性表

Scale	Number of *VMs*=4000									
	Gan-1	Gan-2	Gan-3	Gan-4	Gan-5	Gan-6	Gan-7	Gan-8	Gan-9	Gan-10
Mean (%)	10.75	10.71	10.61	**10.83**	**10.36**	10.40	10.71	**10.62**	10.66	10.52
StDev (%)	14.78	14.55	14.63	**14.98**	**14.16**	14.28	14.93	**14.52**	14.40	14.31

表 8 算法参数设置

Algorithm	*S*	*TG*	w_1	w_2	*MR*	*CR*	*P*	*Y*
HDH-DEPSO	20	20	0.1	0.9	0.5	0.5	0.6	0.4
Discrete-DE	20	20	0.1	0.9	0.5	0.5	–	–
Discrete-PSO	20	20	0.1	0.9	–	–	0.6	0.4

参考文献[26,43,44]中的实验评价指标, 本文使用能耗(EC)、服务水平协议违约率(SLAV)、虚拟机迁移次数(migrations)、活跃服务器数量、能效(EE)这 5 个指标对提出的虚拟机整合方法进行评估.

(a) 服务水平协议违约率(SLAV): 在一个时间片内, 若服务器分配给虚拟机的 CPU 资源量少于其请求的资源量, 则称该虚拟机在这一时间片内超载. 我们用 SLAV 衡量超载的程度, SLAV 在 CloudSim 4.0 中的定义如公式(26)所示, T 为仿真过程中相邻的两个 CloudSim 事件构成的时间片的个数, S_k 为第 k 个时间片内超载的虚拟机数量, $CPU_{k,j}^{Req}$ 和 $CPU_{k,j}^{Allo}$ 分别为 VM_j 在第 k 个时间片的 CPU 资源请求量和实际获取到的 CPU 资源量, tS_k 表示两个 CloudSim 事件之间的时间片长度.

$$SLAV=\frac{\sum_{k=1}^{T}\sum_{j=1}^{S_k}((CPU_{k,j}^{Req}-CPU_{k,j}^{Allo})\cdot tS_k)}{\sum_{k=1}^{T}\sum_{j=1}^{S_k}(CPU_{k,j}^{Req}\cdot tS_k)} \tag{26}$$

(b) 虚拟机迁移次数: 虚拟机迁移次数直接影响数据中心的 QoS, 过多的虚拟机迁移会导致 SLAV 的上升. 在本文的仿真实验中, 我们将虚拟机迁移时的性能退化程度设置为 10%, 即虚拟机在迁移过程中, 仅能获取到自身请求的 90%的 CPU 和 RAM 资源.

(c) 能效(EE): 能效用于衡量服务器在 QoS 和能耗之间的表现. 集群在一定时间内处理的数据量能够直观地反映集群的性能. 我们取它与能耗的比值, 综合地评价服务器的性能与能耗. 其中, 集群处理的数据量以百万指令(MI)为单位, 能耗以焦耳(J)为单位, 如公式(27)所示.

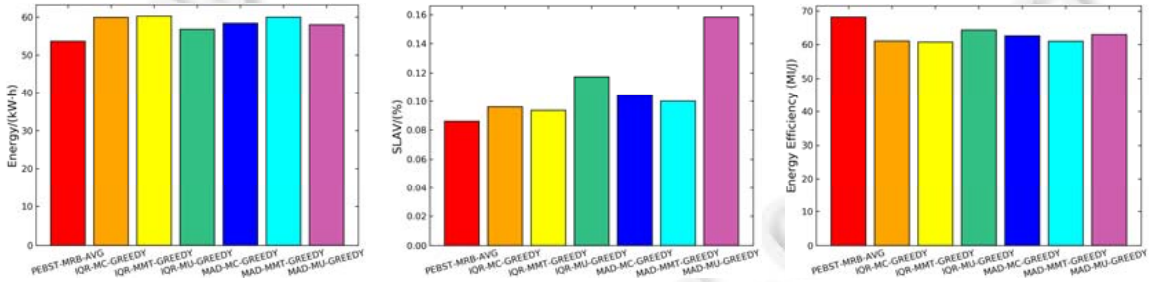
$$EE = \frac{Amount\ of\ data\ processed}{Cluster\ energy\ consumption} \tag{27}$$

为了评估 HSI-VMC 方法的表现，我们对 PEBST-MRB-AVG 策略组合与 HDH-DEPSO 虚拟机放置算法进行了详尽的实验. 实验由两部分组成: (1) 将 PEBST-MRB-AVG 策略组合与文献[44]中的几种调度策略进行对比，分析 PEBST-MRB-AVG 策略组合的能耗和性能表现; (2) 在不同规模的负载数据下，将 HDH-DEPSO 算法与其他几种常见的虚拟机放置算法进行比较，验证 HDH-DEPSO 在能耗、SLAV、能效、时间开销、收敛性等方面的表现.

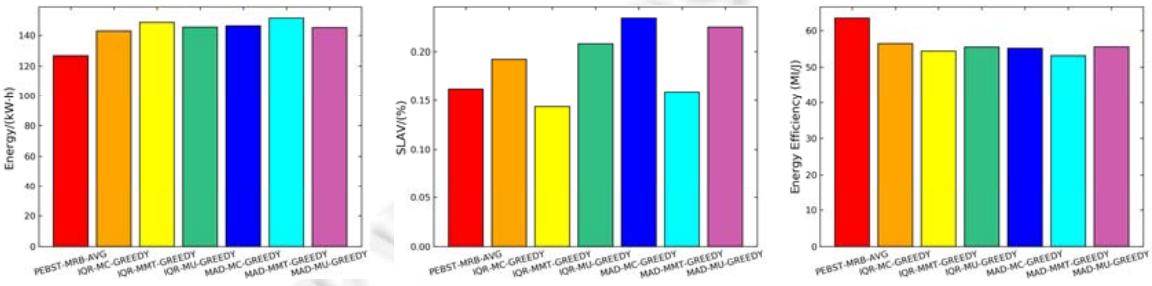
4.2 实验结果

针对 PEBST-MRB-AVG 策略组合的评估，我们选用文献[44]中提出的 IQR，MAD 超载服务器选择策略、MC，MMT，MU 虚拟机选择策略以及贪心欠载服务器处理策略，组合成 6 种不同的虚拟机整合方法进行对比实验. 我们选取 MIX 数据集中负载均值最大的负载(Mix-4)在 Large 规模和 X-Large 规模下的数据进行实验，尽可能突显出不同调度策略各评价指标的差异与变化趋势. 所有策略均采用 MBFD 虚拟机放置算法，以保证放置算法的一致性.

如图 11 所示，在数据中心总能耗方面，本文提出的虚拟机调度策略 PEBST-MRB-AVG 有着不俗的表现. 在 Large 规模实验中, PEBST-MRB-AVG 的能耗相比其他虚拟机调度策略平均降低了 8.84%的能耗, SLAV 优于其他策略，其能效比其他策略至少高出 5.94%. 在 X-Large 规模的实验中，PEBST-MRB-AVG 的能耗和能效表现均为最优，与其他策略相比，其能耗降低了 11%–16%，能效高出 12%–19%，同时能够保持相对较低的 SLAV. 虽然 IQR-MMT-GREEDY 和 MAD-MMT-GREEDY 的 SLAV 值略低于 PEBST-MRB-AVG，但它们的能耗值要比 PEBST-MRB-AVG 高出 15%，且在能效方面的表现不如 PEBST-MRB-AVG. 一方面，PEBST 根据不同类型服务器的峰值效能比进行设定，能够尽可能地使虚拟机迁移后的各服务器保持在效能较高的水平；另一方面, AVG 欠载服务器处理策略能够基于服务器集群 CPU 利用率均值选出合适的欠载服务器进行关断，适量减少空闲服务器，避免了贪心服务器关断策略过度关断服务器导致的过载问题. 与其他策略相比，PEBST-MRB-AVG 能够在保证 QoS 的前提下进一步地降低集群的总能耗.



(a) Large 集群



(b) X-Large 集群

图 11 不同虚拟机调度策略在 Mix-4 负载的对比结果

在虚拟机放置算法方面, 我们把 HSI-VMC 中的 HDH-DEPSO 虚拟机放置算法与另外 7 种启发式虚拟机放置算法做比较, 并分析了算法在各方面的表现. 其中, MBFD, PEAP, FFD 的主要特点是运行时间短, 无须反复迭代, 且均基于某种贪心启发式策略进行虚拟机放置. Discrete-DE, Discrete-PSO, IPSO[28], GAPSO[31]均为群智能算法, 运行时间较长, 其运行结果与算法自身的参数密切相关. 我们选用表 5 中 PlanetLab 20110303–20110420 中 Small, Medium, Large 规模的数据和表 6MIX 中 X-Large 规模的数据以及表 7Gan 中 XX-Large 规模的数据进行实验. 除 IPSO 和 GAPSO 外, 其余放置算法的虚拟机整合策略均采用 HSI-VMC 的 PEBST-MRB-AVG. IPSO 和 GAPSO 的算法迭代次数与 HDH-DEPSO 一致. 对单一规模集群 10 天的仿真结果取均值, 可得表 9–表 13.

在 Small 集群中, 各启发式虚拟机放置算法在能耗方面的差距不大, IPSO 和 GAPSO 与其他几种算法相比表现更优. 在 Medium 集群中, HDH-DEPSO 的能耗表现要优于 IPSO 和 GAPSO 以外的算法, 与 PEAP, MBFD, FFD, Discrete-DE, Discrete-PSO 相比, 分别节省了 27.87%, 4.34%, 24.61%, 2.54%和 6.57%的能耗, 比 IPSO 和 GAPSO 高出 6.52%和 5.06%. 在 Large 集群中, HDH-DEPSO 的能耗表现要优于其他的算法, 其能耗均值与 PEAP, MBFD, FFD 相比, 降低了 29.02%, 15.04%, 32.81%, 比其余 4 种群智能算法至少降低了 7.80%. 在 X-Large 集群中, HDH-DEPSO 和其他算法的能耗差距进一步扩大, 其降幅达到 8.87%–36.59%. 在 XX-Large 集群中, 与其他算法相比, HDH-DEPSO 的节能比例在 8.31%–36.84%之间, 算法之间的能耗差距逐渐趋于稳定. 结果表明: HDH-DEPSO 与其他算法相比, 确实能够在集群规模逐渐增大的情况下找到能耗较小的虚拟机放置方案.

在 QoS 方面, Small 集群下 IPSO 和 GAPSO 的 SLAV 和虚拟机迁移次数远高于其余的算法, HDH-DEPSO 与 MBFD, FFD, Discrete-DE, Discrete-PSO 的 SLAV 相近. 在 Medium 集群中, HDH-DEPSO 的 SLAV 略高于 Discrete-DE 和 Discrete-PSO, 与 PEAP, MBFD, FFD, IPSO, GAPSO 相比降低了 41.94%, 29.25%, 51.87%, 95.40, 95.45%. 在 Large 集群中, HDH-DEPSO 的 SLAV 值要低于除 Discrete-PSO 以外的其他算法, 仅为 IPSO 和 GAPSO 的 5.77%和 5.70%, 与 PEAP, MBFD, FFD, Discrete-DE 相比分别降低了 25.89%, 28.47%, 45.16%, 12.03%. 而在 X-Large 和 XX-Large 集群中, HDH-DEPSO 的 SLAV 稍大于 PEAP, MBFD, Discrete-DE 和 Discrete-PSO, 处于可接受范围. HDH-DEPSO 在虚拟机迁移次数较高的情况下仍然能够保持较低的 SLAV, 是因为 HDH-DEPSO 的服务器开启策略能够根据待放置的虚拟机的资源请求量选出合适的服务器, 并通过几种启发式算法生成较优初始解. 在迭代过程中, HDH-DEPSO 能够通过变异、交叉等操作进一步搜索更好的虚拟机放置方案, 进而缓解服务器由于负载变动而引起的超载问题.

在能效方面, HDH-DEPSO 在 Small 和 Medium 规模集群中的表现并非十分突出. 但从 Large 规模开始, 因为 HDH-DEPSO 能够在保持较低 SLAV 的同时尽可能地降低集群的能耗, 在每一轮的调度中均能获取到适应度值较小的虚拟机放置方案, 所以其能效表现要优于其余的虚拟机放置算法. 在 Large 规模中, HDH-DEPSO 的能效比其他算法高出 9.48%–48.51%. 在 X-Large 规模中, HDH-DEPSO 与其余算法的能效差距进一步扩大, 比其他算法高出 9.68%–57.48%. 在 XX-Large 规模中, HDH-DEPSO 的能效表现保持稳定, 比其他算法高出 8.92%–58.17%.

在时间开销方面, 由于群智能算法需要通过多次迭代更新群体, HDH-DEPSO, Discrete-DE, Discrete-PSO, IPSO, GAPSO 的时间开销高于其他启发式算法. 其中, HDH-DEPSO, Discrete-DE, Discrete-PSO 算法每次迭代都需要对群体进行排序, 且它们需要多次调用 MBFD 算法重新放置算法过程中放置冲突的虚拟机, 因此算法的时间开销要高于 IPSO 和 GAPSO. 考虑到现有云数据中心的调度场景需求和虚拟机调度的时间间隔, 我们采用多线程技术进一步改良了 HDH-DEPSO, 在大幅度减少算法运行时间的同时提升了算法的运行效率, 使得 HDH-DEPSO 的时间开销能够保持在可接受的范围内. 表 9–表 13 展示了线程数量为 10 时, HDH-DEPSO, Discrete-DE, Discrete-PSO 这 3 种算法的时间开销.

表 9 Small 集群-虚拟机放置算法对比结果均值表

Algorithm	Energy (kW·h)	Migrations	SLAV (%)	EE (MI/J)	Overhead (ms)
HDH-DEPSO	12.45	415	0.026 0	50.83	5.29
PEAP	15.68	503	0.043 6	41.67	1.15
MBFD	10.98	**119**	**0.004 6**	57.08	**0.74**
FFD	13.19	242	0.013 5	48.09	0.75
Discrete-DE	12.34	328	0.016 8	51.23	2.42
Discrete-PSO	12.81	373	0.017 7	49.37	2.85
IPSO	**9.55**	11 350	0.902 4	**65.19**	15.77
GAPSO	9.69	10 734	0.860 5	62.16	37.64

表 10 Medium 集群-虚拟机放置算法对比结果均值表

Algorithm	Energy (kW·h)	Migrations	SLAV (%)	EE (MI/J)	Overhead (ms)
HDH-DEPSO	36.39	4 362	0.067 0	58.9	83.84
PEAP	50.45	3 971	0.115 4	42.95	2.79
MBFD	38.04	4 512	0.094 7	57.03	2.6
FFD	48.27	4 875	0.139 2	44.77	**2.53**
Discrete-DE	37.34	4 035	0.073 8	57.74	54.11
Discrete-PSO	38.95	**3 422**	**0.055 5**	55.29	44.96
IPSO	**34.39**	45 128	1.458	**60.57**	53.59
GAPSO	34.93	45 115	1.475 3	58.74	114.01

表 11 Large 集群-虚拟机放置算法对比结果均值表

Algorithm	Energy (kW·h)	Migrations	SLAV (%)	EE (MI/J)	Overhead (ms)
HDH-DEPSO	**59.11**	10 379	0.090 7	**64.41**	254.26
PEAP	82.02	8 196	0.122 4	45.82	4.65
MBFD	69.4	9 449	0.119 7	54.79	5.76
FFD	86.52	9 107	0.159 2	43.37	**4.28**
Discrete-DE	64.85	9 149	0.103 1	58.83	159.89
Discrete-PSO	68.07	**5 855**	**0.058 1**	56.06	113.7
IPSO	64.11	81 528	1.571 7	57.41	105.88
GAPSO	65.88	82 185	1.592 4	55.28	244.51

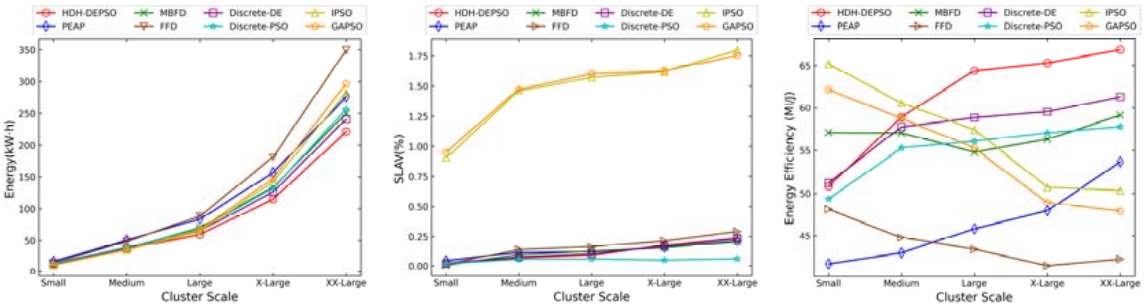
表 12 X-Large 集群-虚拟机放置算法对比结果均值表

Algorithm	Energy (kW·h)	Migrations	SLAV (%)	EE (MI/J)	Overhead (ms)
HDH-DEPSO	**114.6**	32 068	0.178 6	**65.29**	1201.44
PEAP	156.45	19 103	0.159 9	47.93	9.22
MBFD	133.22	23 621	0.155 6	56.29	15.89
FFD	180.72	20 844	0.214 7	41.46	**8.97**
Discrete-DE	125.76	26 192	0.170 1	59.53	842.85
Discrete-PSO	131.67	**10 070**	**0.048**	57.05	389.61
IPSO	142.24	170 895	1.617 4	50.73	247.06
GAPSO	147.44	172 338	1.622 4	48.91	594.95

表 13 XX-Large 集群-虚拟机放置算法对比结果均值表

Algorithm	Energy (kW·h)	Migrations	SLAV (%)	EE (MI/J)	Overhead (ms)
HDH-DEPSO	**220.70**	69 655	0.237 6	**66.78**	4 079.7
PEAP	275.32	46 776	0.213 8	53.71	18.57
MBFD	249.82	58 056	0.209 8	59.12	45.44
FFD	349.41	51 801	0.289 7	42.22	**16.61**
Discrete-DE	240.71	62 132	0.233 6	61.31	3 853.56
Discrete-PSO	256.38	**23 244**	**0.059 6**	57.79	1 080.5
IPSO	281.99	330 611	1.797 3	50.35	1 178.3
GAPSO	296.53	332 648	1.752 2	47.88	3 678.46

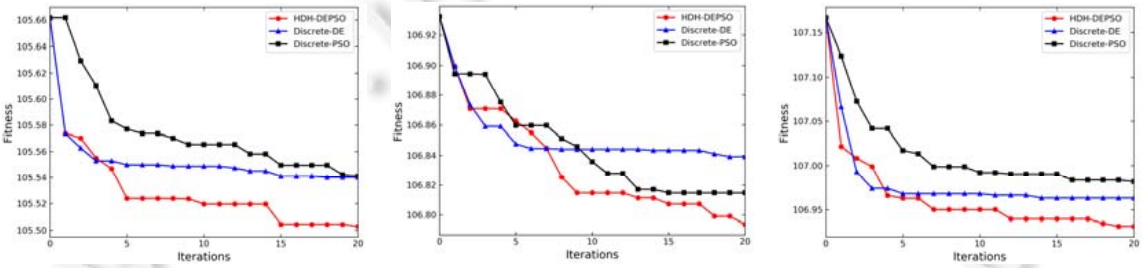
图 12 展示了各虚拟机放置算法在不同规模下能耗、SLAV、能效的变化趋势. 显然, 随着集群规模的增长, 问题的搜索空间也在逐步扩大, 要搜索出更优的虚拟机放置方案是困难的. 基于贪心策略的启发式算法易陷入局部最优, 难以获取较优的虚拟机放置方案. 从图 12 中可以看出: HDH-DEPSO 在 Small 集群中的表现与其他虚拟机放置算法相近; 但随着集群规模的增长, HDH-DEPSO 无论是在能耗还是 SLA*V* 上都有着明显的优势, 能够在能耗和 QoS 之间找到相对较好的平衡点. 因此, HDH-DEPSO 的能效表现优于其他的算法. IPSO 和 GAPSO 虽然在能耗方面上有着不错的表现, 但其算法的调度范围是数据中心中的所有虚拟机, 虚拟机迁移次数激增易导致 SLAV 上升, 影响数据中心的 QoS. 因此, 它们的能效会随着规模的增大而降低.



(a) 能耗均值 (b) SLAV 均值 (c) 能效均值

图 12 能耗均值、SLAV 均值、能效均值在不同规模集群下的变化曲线

为进一步对比 HDH-DEPSO, Discrete-DE, Discrete-PSO 的收敛能力, 我们收集了 Mix 数据集中负载均值为最小值、中位数、最大值的负载(Mix-10, Mix-5, Mix-4)在 X-Large 规模中, 3 种算法第 1 次调度的适应度值变化数据, 如图 13 所示. HDH-DEPSO 在迭代后所得到的虚拟机放置方案的适应度值总是最小的, 说明 HDH-DEPSO 的收敛能力确实比 Discrete-DE 和 Discrete-PSO 更强. HDH-DEPSO 结合了 DE 和 PSO, 既具备 PSO 较强的局部收敛能力, 又能通过 DE 的交叉和变异操作, 对解向量进行适度的扰动, 能够保持群体的多样性, 防止搜索过程的早熟收敛, 避免陷入局部最优, 具备较强的寻优能力. 结合本文给出的服务器开启策略, HDH-DEPSO 与 Discrete-DE 和 Discrete-PSO 相比, 能够在庞大的集群中找出更加节能、高效的虚拟机调度方案.



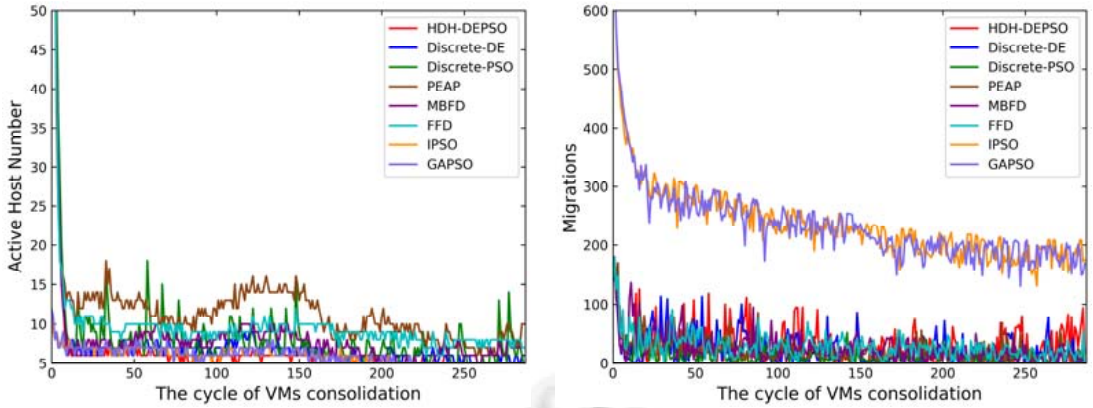
(a) Mix-10 (b) Mix-5 (c) Mix-4

图 13 不同算法在 Mix-10、Mix-5、Mix-4 负载下的收敛性分析

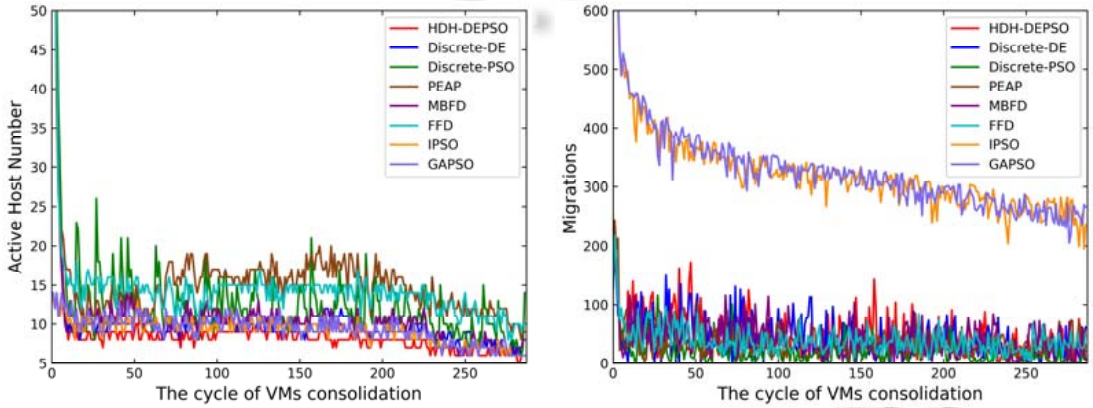
图 14 展示了每一轮虚拟机调度中, 剩余的活跃服务器数量和虚拟机迁移次数. 在每一轮调度中, 虚拟机整合方法应该尽可能地缩减活跃服务器的数量, 在减少服务器资源碎片的同时, 降低服务器的实时功耗. 我们选取 Large 规模中负载均值为最小值、中位数、最大值的 PlanetLab 工作负载 20110322, 20110412, 20110303 作为样本, 来分析不同算法在一天时间内多次调度后的活跃服务器与虚拟机迁移次数的变化情况. 从第 5 次调度开始, 各负载数据下不同算法的活跃服务器数量开始出现一定的差距. PEAP, Discrete-PSO, FFD, MBFD 的表现相近, 其活跃服务器数量普遍偏高但虚拟机迁移次数相对较少. Discrete-DE, IPSO, GAPSO 的活跃服务器数量在调度过程基本维持在 10–15 台之间, 小于上述的几种算法. HDH-DEPSO 在活跃服务器数量上能够维持在 10 台以下, 说明 HSI-VMC 在一定程度上减少了服务器的资源碎片, 并有效地降低服务器集群的能耗.

PEAP, FFD, MBFD 在虚拟机放置过程中往往会将虚拟机放置在启发式规则下的最优服务器上. 然而在批量虚拟机放置的过程中, 虚拟机放置的先后顺序会对放置时各服务器的资源状态和启发式规则的判定产生影响, 容易陷入局部最优, 因而难以进一步减少活跃服务器的数量. 而由于活跃服务器数量的增加, 单个服务器上的虚拟机数量更少, Host 负载波动的减少同样会减少超载和欠载服务器的数量, 因而虚拟机迁移次数相对较小. Discrete-PSO 在解的搜索过程中缺乏对解向量的大幅度扰动, 在迭代后期易陷入局部最优, 无法搜索到功耗更低的虚拟机放置方案. 因此, 在负载波动较大时, 其活跃服务器数量会出现峰值(如图 14(a)–(c)左图的

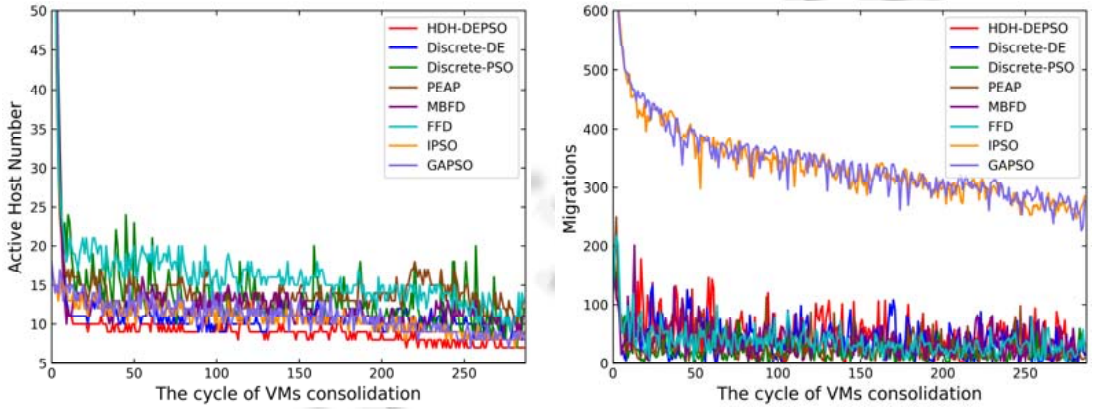
绿线所示). IPSO 和 GAPSO 在各次虚拟机调度中虽然能够保持较小的活跃服务器数量, 但其虚拟机迁移次数远高于其余的算法, 容易影响 QoS, 并不适用于集群规模太大的虚拟机调度. Discrete-DE 和 HDH-DEPSO 在活跃服务器数量上的表现非常接近, 在每次虚拟机调度中都能维持较低的活跃服务器数量. 为了减少活跃服务器数量, 它们的虚拟机迁移次数会有所增加, 但不会对 QoS 产生太大的影响, 因此具有较高的能效值.



(a) 20110322



(b) 20110412



(c) 20110303

图 14　活跃服务器数量与虚拟机迁移次数在 PlanetLab 负载数据随虚拟机调度周期的变化

5 结 论

本文首先讨论了数据中心节能的重要性和必要性，通过分析现有的群智能虚拟机调度算法，发现了单一群智能虚拟机调度算法存在的问题. 进一步地，我们给出了虚拟机整合的问题定义及优化目标，并提出了一种节能且高效的 HSI-VMC 方法，包括 PEBST 超载服务器检测策略与 MRB 待迁移虚拟机选择策略、目标服务器选择策略、HDH-DEPSO 虚拟机放置算法以及 AVG 欠载服务器处理策略. 最后，我们把 HSI-VMC 方法与 GAPSO, IPSO 等虚拟机整合算法进行对比，评估了 HSI-VMC 方法的表现. 实验结果表明：与几种同类型的虚拟机调度算法相比，HSI-VMC 方法在 Large 规模以上的集群环境的能耗降幅至少为 7.8%，对能效的提升可达 8.92%−9.68%. 同时，该方法兼顾了 SLAV 和虚拟机迁移次数这两个重要的 QoS 指标，保证了虚拟机和服务器的性能.

HSI-VMC 方法能够有效地降低服务器集群的能耗，但是仍存在一定的改良空间. 随着集群规模的增长，HSI-VMC 的虚拟机迁移次数也会随之上升，过多的虚拟机迁移可能会影响虚拟机的性能. 因此，在未来的工作中，我们会结合机器学习对虚拟机的负载数据进行预测，考虑负载的变化趋势进行虚拟机调度，进一步地降低虚拟机的迁移次数.

致谢 在此，我们向对相关项目进行和论文撰写工作做出贡献和提出宝贵意见的学者，特别是在华南理工大学计算机科学与工程学院先进计算体系结构团队工作的老师和同学表示感谢.

References:

[1] Research: There are now close to 400 hyper-scale data centers in the world. 2020. https://www.datacenterknowledge.com/cloud/research-there-are-now-close-400-hyper-scale-data-centers-world

[2] Networking CV. Cisco Global Cloud Index: Forecast and Methodology 2015−2020. White Paper, 2019.

[3] Avgerinou M, Bertoldi P, Castellazzi L. Trends in data centre energy consumption under the European code of conduct for data centre energy efficiency. Energies, 2017, 10(10): Artical No.1470.

[4] Lin WW, Qi DY. Survey of resource scheduling in cloud computing. Computer Science, 2012, 39(10): 1−6 (in Chinese with English abstract).

[5] Li DH, Zhao JC, Cui HM, Feng XB. Modeling the impact of DVFS on performance of applications in datacenter. Ruan Jian Xue Bao/Journal of Software, 2017, 28(4): 845−859 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5194.htm [doi: 10.13328/j.cnki.jos.005194]

[6] Stavrinides GL, Karatza HD. An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations. Future Generation Computer Systems, 2019, 96: 216−226.

[7] Shirvani MH, Rahmani AM, Sahafi A. A survey study on virtual machine migration and server consolidation techniques in DVFS-enabled cloud datacenter: Taxonomy and challenges. Journal of King Saud University—Computer and Information Sciences, 2020, 32(3): 267−286.

[8] Lin WW, Liu B, Zhu LC, Qi DY. CSP-based resource allocation model and algorithms for energy-efficient cloud computing. Journal on Communications, 2013, 12(1): 33−41 (in Chinese with English abstract).

[9] Xu SY, Lin WW, Wang ZJ. Virtual machine placement algorithm based on peak workload characteristics. Ruan Jian Xue Bao/Journal of Software, 2016, 27(7): 1876−1887 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4918.htm [doi: 10.13328/j.cnki.jos.004918]

[10] Alahmadi A, Alnowiser A, Zhu MM, Che D, Ghodous P. Enhanced first-fit decreasing algorithm for energy-aware job scheduling in cloud. In: Proc. of the Int'l Conf. on Computational Science and Computational Intelligence. IEEE, 2014. 69−74.

[11] Kumar S, Mishra A. Application of min-min and max-min algorithm for task scheduling in cloud environment under time shared and space shared vm models. Int'l Journal of Computing Academic Research (IJCAR), 2015, 4(6): 182−190.

[12] Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems, 2012, 28(5): 755−768.

[13] Koza JR. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, 1992.

[14] Storn R, Price K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997, 11(4): 341−359.

[15] Kennedy J, Eberhart R. Particle swarm optimization. In: Proc. of the Int'l Conf. on Neural Networks (ICNN 1995). IEEE, 1995. 1942−1948.

[16] Dorigo M, Di Caro G. Ant colony optimization: A new meta-heuristic. In: Proc. of the Congress on Evolutionary Computation (CEC'99). IEEE, 1999. 1470−1477.

[17] Karaboga D. An idea based on honey bee swarm for numerical optimization. Technical Report, tr06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[18] Agharazi H, Kolacinski RM, Theeranaew W, Loparo KA. A swarm intelligence-based approach to anomaly detection of dynamic systems. Swarm and Evolutionary Computation, 2019, 44: 806−827.

[19] Ari AAA, Gueroui A, Titouna C, Thiare O, Aliouat Z. Resource allocation scheme for 5G C-RAN: A swarm intelligence based approach. Computer Networks, 2019, 165: Article No.106957.

[20] Zhao X, Wang C, Su J, Wang J. Research and application based on the swarm intelligence algorithm and artificial intelligence for wind farm decision system. Renewable Energy, 2019, 134: 681−697.

[21] Kang K, Bae C, Yeung HWF, Chung YY. A hybrid gravitational search algorithm with swarm intelligence and deep convolutional feature for object tracking optimization. Applied Soft Computing, 2018, 66: 319−329.

[22] Logesh R, Subramaniyaswamy V, Vijayakumar V, Gao XZ, Indragandhi V. A hybrid quantum-induced swarm intelligence clustering for the urban trip recommendation in smart city. Future Generation Computer Systems, 2018, 83: 653−673.

[23] Sato M, Fukuyama Y, Iizaka T, Matsui T. Total optimization of energy networks in a smart city by multi-swarm differential evolutionary particle swarm optimization. IEEE Trans. on Sustainable Energy, 2018, 10(4): 2186−2200.

[24] Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 2011, 41(1): 23−50.

[25] Yousefipour A, Rahmani AM, Jahanshahi M. Energy and cost-aware virtual machine consolidation in cloud computing. Software: Practice and Experience, 2018, 48(10): 1758−1774.

[26] Li Z, Yu X, Yu L, Guo S, Chang V. Energy-efficient and quality-aware VM consolidation method. Future Generation Computer Systems, 2020, 102: 789−809.

[27] Jiang Y, Wang J, Shi J, *et al*. Network-aware virtual machine migration based on gene aggregation genetic algorithm. Mobile Networks and Applications, 2020, 25(4): 1457−1468.

[28] Wang S, Liu Z, Zheng Z, Sun Q, Yang F. Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers. In: Proc. of the Int'l Conf. on Parallel and Distributed Systems. IEEE, 2013. 102−109.

[29] Ma T, Xu C, Zhou Z, Kuang X, Zhong L. SE-PSO: Resource scheduling strategy for multimedia cloud platform based on security enhanced virtual migration. In: Proc. of the 15th Int'l Wireless Communications & Mobile Computing Conf. (IWCMC). IEEE, 2019. 650−655.

[30] Yan J, Zhang H, Xu H, Zhang Z. Discrete PSO-based workload optimization in virtual machine placement. Personal and Ubiquitous Computing, 2018, 22(3): 589−596.

[31] Sharma NK, Reddy GRM. Multi-objective energy efficient virtual machines allocation at the cloud data center. IEEE Trans. on Services Computing, 2016, 12(1): 158−171.

[32] Meshkati J, Safi-Esfahani F. Energy-aware resource utilization based on particle swarm optimization and artificial bee colony algorithms in cloud computing. The Journal of Supercomputing, 2019, 75(5): 2455−2496.

[33] Yan W, Chen J, Li L. A power-aware ACO algorithm for the cloud computing platform. In: Proc. of the 4th Int'l Conf. on Communication and Information Processing. 2018. 1−6.

[34] Malekloo MH, Kara N, El Barachi M. An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments. Sustainable Computing: Informatics and Systems, 2018, 17: 9–24.

[35] Farahnakian F, Ashraf A, Pahikkala T, Liljeberg P, Plosila J, Porres I, Tenhunen H. Using ant colony system to consolidate VMs for green cloud computing. IEEE Trans. on Services Computing, 2014, 8(2): 187–198.

[36] Ragmani A, Elomri A, Abghour N, *et al*. FACO: A hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing. Journal of Ambient Intelligence and Humanized Computing, 2020, 11(10): 3975–3987.

[37] Xiao H, Hu Z, Li K. Multi-objective VM consolidation based on thresholds and ant colony system in cloud computing. IEEE Access, 2019, 7: 53441–53453.

[38] Li Z, Yan C, Yu L, Yu X. Energy-aware and multi-resource overload probability constraint-based virtual machine dynamic consolidation method. Future Generation Computer Systems, 2018, 80: 139–156.

[39] Jiang J, Feng Y, Zhao J, Li K. DataABC: A fast ABC based energy-efficient live VM consolidation policy with data-intensive energy evaluation model. Future Generation Computer Systems, 2017, 74: 132–141.

[40] Yavari M, Rahbar AG, Fathi MH. Temperature and energy-aware consolidation algorithms in cloud computing. Journal of Cloud Computing, 2019, 8(1): 1–16.

[41] Abdel-Basset M, Abdle-Fatah L, Sangaiah AK. An improved Lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment. Cluster Computing, 2019, 22(4): 8319–8334.

[42] Al-Moalmi A, Luo J, Salah A, Li K. Optimal virtual machine placement based on grey wolf optimization. Electronics, 2019, 8(3): 283.

[43] Lin W, Wu W, He L. An on-line virtual machine consolidation strategy for dual improvement in performance and energy conservation of server clusters in cloud data centers. IEEE Trans. on Services Computing, 2019, 15(2): 766–777.

[44] 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. Concurrency and Computation: Practice and Experience, 2012, 24(13): 1397–1420.

附中文参考文献:

[4] 林伟伟, 齐德昱. 云计算资源调度研究综述. 计算机科学, 2012, 39(10): 1–6.

[5] 李登辉, 赵家程, 崔慧敏, 冯晓兵. 数据中心中 DVFS 对程序性能影响模型的设计. 软件学报, 2017, 28(4): 845–859. http://www.jos.org.cn/1000-9825/5194.htm [doi: 10.13328/j.cnki.jos.005194]

[8] 林伟伟, 刘波, 朱良昌, 齐德昱. 基于 CSP 的能耗高效云计算资源调度模型与算法. 通信学报, 2013, 12(1): 33–41.

[9] 徐思尧, 林伟伟, 王子骏. 基于负载高峰特征的虚拟机放置算法. 软件学报, 2016, 27(7): 1876–1887. http://www.jos.org.cn/1000-9825/4918.htm [doi: 10.13328/j.cnki.jos.004918]

李俊祺(1996－), 男, 硕士生, 主要研究领域为群智能算法, 数据中心虚拟机调度.

石方(1993－), 女, 博士生, CCF 学生会员, 主要研究领域为资源调度, 智能算法.

林伟伟(1980－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为云计算能耗建模和调度优化, 大数据架构和分析算法, AI 应用技术.

李克勤(1963－), 男, 博士, 教授, 博士生导师, 主要研究领域为雾计算和移动边缘计算, 高能效计算和通信, 物联网和信息物理系统, 异构计算系统, 大数据计算, CPU-GPU 混合协同计算, 智能计算.