基于多目标优化的大规模Hadoop集群虚拟机放置

文佳, 吴舒霞, 于正欣, 苗旺, 陈哲毅

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.241200020

基于多目标优化的大规模 Hadoop 集群虚拟机放置

文　佳[1,2,3]　吴舒霞[1,2,3]　于正欣[4]　苗　旺[5]　陈哲毅[1,2,3]
1 福州大学计算机与大数据学院　福州 350116
2 大数据智能教育部工程研究中心　福州 350002
3 福建省网络计算与智能信息处理重点实验室(福州大学)　福州 350116
4 兰卡斯特大学计算与通信学院　英国 兰卡斯特 LA1 4YW
5 埃克塞特大学计算机科学系　英国 埃克塞特 EX4 4QF
(1448341761@qq.com)

摘　要　虚拟化技术已成为云计算快速发展的核心支撑。Hadoop 作为一种广泛应用于云环境中的分布式框架,其集群性能通常受限于低下的资源管理效率。随着数据量与集群规模的不断增大,如何高效优化虚拟机放置进而降低 Hadoop 集群能耗、提升资源利用率和缩短文件访问延迟已成为一个极具挑战的难题。对此,提出了新型的面向大规模 Hadoop 集群虚拟机放置的可变长度双染色体多目标优化(Multi-objective Optimization with Variable Length Double chromosome,MO-VLD)方法。首先,通过结合可变长度染色体与非支配排序遗传算法(Non-dominated Sorting Genetic Algorithm-III,NSGA-III),设计了双染色体结构。接着,引入两阶段交叉与变异操作以增强解空间探索的多样性。基于谷歌集群真实运行数据集的大量实验表明,MO-VLD 方法能够有效应对动态的资源需求并提升 Hadoop 集群的资源管理效率。相比于基准方法,MO-VLD 方法在能耗、资源利用率和文件访问延迟方面均展现出更加优越的性能。

关键词:云计算;Hadoop;虚拟机放置;多目标优化;遗传算法
中图分类号　TP393

Multi-objective Optimization for Virtual Machine Placement in Large-scale Hadoop Cluster

WEN Jia[1,2,3],WU Shuxia[1,2,3],YU Zhengxin[4],MIAO Wang[5] and CHEN Zheyi[1,2,3]
1 College of Computer and Data Science,Fuzhou University,Fuzhou 350116,China
2 Key Laboratory of Spatial Data Mining & Information Sharing,Ministry of Education,Fuzhou 350002,China
3 Fujian Provincial Key Laboratory of Networking Computing and Intelligent Information Processing,Fuzhou 350116,China
4 School of Computing and Communications,Lancaster University,Lancaster LA1 4YW,UK
5 Department of Computer Science,University of Exeter,Exeter EX4 4QF,UK

Abstract　Virtualization technology has become the core support for the rapid development of cloud computing. As a popular distributed framework in cloud environments,the performance of the Hadoop cluster is usually limited by the low efficiency of resource management. With the increasing data volume and cluster scale,it is challenging to efficiently optimize Virtual Machine (VM) placement in the Hadoop cluster to reduce energy consumption,increase resource utilization,and lessen file access latency. To address this important challenge,this paper proposes a novel Multi-objective Optimization with Variable Length Double chromosome(MO-VLD) method for VM placement in the large-scale Hadoop cluster. Firstly,a double chromosome structure is designed by combining the variable length chromosome with NSGA-III. Next,two-stage crossover and mutation operations are introduced to enhance the exploration diversity of solution space. Using the real-world runtime datasets of the Google cluster,extensive simulation experiments demonstrate that the proposed MO-VLD method can effectively handle the dynamic resource demands and improve the resource management efficiency of the Hadoop cluster. Compared to benchmark methods,the MO-VLD method shows superior performance in terms of energy consumption,resource utilization,and file access latency.

到稿日期:2024-12-02　返修日期:2025-03-18
基金项目:国家自然科学基金(62202103);福建省自然科学基金杰出青年基金(2025J010020);中央引导地方科技发展资金项目(2022L3004);福建省科技经济融合服务平台(2023XRH001);福厦泉国家自主创新示范区协同创新平台项目(2022FX5)
This work was supported by the National Natural Science Foundation of China(62202103),Natural Science Foundation of Fujian Province for Distinguished Young Fund(2025J010020),Central Funds Guiding the Local Science and Technology Development(2022L3004),Fujian Province Technology and Economy Integration Service Platform(2023XRH001) and Fuzhou-Xiamen-Quanzhou National Independent Innovation Demonstration Zone Collaborative Innovation Platform(2022FX5).
通信作者:陈哲毅(z.chen@fzu.edu.cn)

Keywords Cloud computing, Hadoop, Virtual machine placement, Multi-objective optimization, Genetic algorithm

1 引言

随着云计算的快速发展，虚拟化技术已成为云数据中心的核心支撑。通过接入云计算服务，用户可按需访问和利用资源（如 CPU/GPU、内存和存储等）[1]。这种按需服务的方式显著提高了云数据中心的资源利用率，同时也降低了其运营成本。因此，越来越多的用户将计算任务上传至云端，以应对日益增长的数据处理需求。

作为一种经典的分布式框架，Hadoop 有着优秀的扩展性与灵活性，已被广泛应用于云环境中[2]。作为 Hadoop 的核心技术之一，分布式文件系统（Hadoop Distributed File System，HDFS）将文件切分为多个数据块并将其存储在多个数据节点上，进而保证数据的高可用性和容错性[3]。作为 Hadoop的另一核心技术，MapReduce 将计算任务调度至数据块所在的节点上进行处理，以最大化数据本地性，缩短数据传输延迟[4]。然而，随着数据量和集群规模的不断增长，如何高效优化虚拟机放置以提升集群资源利用率和任务处理效率，成为 Hadoop 集群性能优化的核心问题。虚拟机放置策略直接影响 Hadoop 集群能耗、资源利用率和文件访问延迟等关键性能指标。因此，如何在大规模动态的 Hadoop 集群中高效地管理资源，成为一个开放性的研究热点。

传统的虚拟机放置大多采用基于遗传算法的策略[5-6]，其在面对小规模集群时表现出良好的性能，但在面对大规模集群和多变负载时暴露出了局限性，无法很好地适应动态变化的资源需求。为解决该问题，一些研究提出多目标优化算法，如多目标粒子群优化（Multi-Objective Particle Swarm Optimization，MOPSO）算法[7]与多目标模拟退火（Multi-Objective Simulated Annealing，MOSA）算法[8]。具体而言，MOPSO 算法通过模拟群体行为来进行全局搜索，在复杂环境下易陷入局部最优且对超参数敏感，因此策略优化过程难以控制。MOSA 算法通过模拟物理退火过程中的能量变化来寻找全局最优解，能够避免陷入局部最优，但收敛速度较慢，且在处理多目标优化问题时会造成系统开销过大。此外，上述算法无法有效处理虚拟机放置问题中的多维异构资源。对此，本文提出了面向大规模 Hadoop 集群虚拟机放置的可变长度双染色体多目标优化（MO-VLD）方法。本文的主要贡献有：

1）设计了新型的面向大规模 Hadoop 集群虚拟机放置的系统模型，将能耗、资源利用率与文件访问延迟视为多优化目标。特别地，所提出的系统模型充分考虑资源需求的动态变化，确保 Hadoop 集群在高负载情况下仍能保持良好性能。

2）提出了可变长度双染色体多目标优化（MO-VLD）方法，通过结合可变长度染色体与 NSGA-III 中的双染色体结构，显著提升了 MO-VLD 方法对多目标优化问题的处理能力。具体而言，首先通过两阶段单点交叉与变异操作和块替换机制增加解空间的探索多样性。接着，利用参考点关联和非支配排序进行最优解的筛选。最后，通过归一化处理以提高解的适应性，使得虚拟机放置策略能够有效应对 Hadoop 集群中的动态资源需求。

3）基于谷歌集群真实运行数据集，通过大量实验验证了 MO-VLD 方法的可行性和有效性。实验结果表明，在不同场景下，相比于其他基准方法，MO-VLD 方法在能耗、资源利用率与文件访问延迟等关键指标上均取得了更加优越的性能。

本文其余部分组织如下：第 2 章分析了虚拟机放置的相关工作；第 3 章描述了虚拟机放置模型并对多目标优化问题进行了形式化定义；第 4 章对 MO-VLD 方法进行了详细介绍；第 5 章评估了 MO-VLD 方法的性能并与其他基准方法进行了对比实验；最后总结并展望了未来研究方法。

2 相关工作

面对虚拟机放置问题中多维异构资源的挑战，一些研究提出利用启发式方法来应对。例如，Srivastava 等[9]提出了一种基于双染色体的遗传算法，旨在减少活跃服务器数量以降低资源浪费。Yarahmadi 等[10]设计了一种改进的基于遗传算法的虚拟机分配策略，以提升资源利用率和降低能耗。Swain 等[11]利用第 3 代非支配排序遗传算法（Non-dominated Sorting Genetic Algorithm-III，NSGA-III）整合多种资源约束，进而提升资源利用率和减少能耗。Tang 等[12]提出了一种混合遗传算法，通过优化能耗和资源利用率，以优化数据中心的虚拟机放置问题。Gopu 等[13]研究了在分布式云环境下的虚拟机放置问题，以减少资源浪费、功耗和传播时间。上述研究通常针对的是小规模集群，难以适应大规模集群中具有动态资源需求的复杂虚拟机放置问题。

针对 Hadoop 集群环境，一些研究工作探讨了虚拟机放置策略对其集群性能的影响。例如，Conejero 等[14]研究了多租户 Hadoop 集群中的虚拟机放置问题，以优化能耗和提升资源利用率。Hedayati 等[15]对 Hadoop 集群中现有调度算法进行了分类和对比。Guerrero 等[16]研究了 Hadoop 集群中的虚拟机分配和副本放置问题，以提升资源利用率和可用性。Marquez 等[17]提出了一种异构云环境下的遗传编程框架，以减少工作负载竞争。针对虚拟机放置策略，Li 等[18]与 Ghazali 等[19]分别侧重于提升 Hadoop 集群的资源利用率和数据本地性。Zhang 等[20]通过分支定界算法获取虚拟机放置的最优解，以缩短数据访问延迟。Naik 等[21]提出了一种针对 Hadoop 集群的负载均衡策略，通过动态调整数据分布来优化资源利用率。Miriam 等[22]设计了一种基于多目标进化的虚拟机放置方法，旨在降低 Hadoop 集群能耗并提高资源利用率。现有研究工作在优化虚拟机放置和提升 Hadoop 集群性能方面取得了一些进展，但如何在资源利用率、能耗和文件访问延迟之间实现高效平衡仍是一个开放性的挑战。

不同于现有研究工作，针对大规模 Hadoop 集群场景，本文所提出的 MO-VLD 在充分考虑资源需求动态性的基础上，通过结合可变长度双染色体和改进的遗传算法，对能耗、资源利用率和文件访问延迟实现了有效的多目标优化。

3 系统模型与问题定义

3.1 系统概览

如图 1 所示，本文的系统模型包含多个机架，每个机架内

配置多个物理机(Physics Machines,PMs),每个物理机上运行着多个虚拟机(Virtual Machines,VMs)。机架之间通过核心交换机互联,机架内部的 PMs 通过交换机互联。Hadoop 集群通过交换机进行数据块传输与文件副本复制,从而实现跨机架的数据同步与负载均衡。具体地,在 Hadoop 集群中有 M 台物理机,记为 $PM_i(i=1,2,\cdots,M)$,其资源配置记为四元组 $C_{pm_i}=(C_{cpu_i},C_{mem_i},C_{io_i},C_{net_i})$,分别表示该 PM 的 CPU、内存、磁盘 I/O 和网络带宽。每个 PM_i 运行不同数量的 VMs,其共享 PM_i 的资源。每个 $VM_j(j=1,2,\cdots,N)$的资源使用情况记为四元组 $U_{vm_j}=(U_{cpu_j},U_{mem_j},U_{io_j},U_{net_j})$。虚拟机所分配到的资源受限于物理机的资源配置,表示为 $U_{mem_j}\leqslant C_{mem_i}$,$U_{io_j}\leqslant C_{io_i}$,$U_{net_j}\leqslant C_{net_i}$。

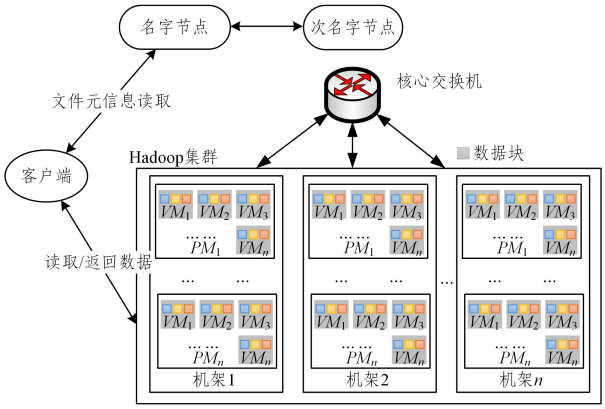


图 1　面向大规模 Hadoop 集群的虚拟机放置

Fig. 1　Virtual machine placement for large-scale Hadoop cluster

为避免物理机的资源不被过度消耗,虚拟机所分配资源和虚拟化软件开销之和不得超过物理机的资源配置,定义为:

$$\sum_{pm_i}U_{pm_i}+U_{hyp_{pm_i}}\leqslant C_{pm_i},\forall\, pm_i\in PM \tag{1}$$

其中,$\sum_{pm_i}U_{pm_i}$ 表示 PM_i 上所有虚拟机的资源消耗之和;U_{hyp_i} 表示 PM_i 上虚拟化软件的资源消耗。同时,VM_j 的资源消耗不能超过其所分配到的资源,定义为:

$$U_{vm_j}\leqslant C_{vm_j},\forall\, vm_j\in VM \tag{2}$$

在 Hadoop 集群中,文件被切分为多个数据块 $B_k(k=1,2,\cdots,K)$,其副本被存储在不同虚拟机上。因此,文件可表示为块的连接,定义为:

$$f_u=fb_0^u\parallel fb_1^u\parallel\cdots\parallel fb_{fC_{fu}-1}^u \tag{3}$$

文件的大小 $fSize_{fu}$ 与数据块的大小 $fbSize_{fu}$ 决定了数据块的数量,定义为:

$$fC_{fu}=\frac{fSize_{fu}}{fbSizefu} \tag{4}$$

为了提高数据可用性与系统容错性,每个数据块会在不同虚拟机上存储多个副本 R_k。文件 f_u 的每个数据块 fb_x^u 会有多个副本,定义为:

$$fb_x^u=\{fb_0^u,fb_1^u,\cdots,fb_{fR_{fu}-1}^u\} \tag{5}$$

其中,fR_{fu} 表示文件 f_u 的副本数量。

PM_i 不能同时存储同一数据块的多个副本,以避免存储资源浪费,该约束表示为:

$$x_{i,k_1}+x_{i,k_2}\leqslant 1,\forall\, i\in PM,\forall\, k_1\neq k_2\in B \tag{6}$$

3.2　能耗模型

PM_i 的能耗受 CPU、网络带宽和磁盘 I/O 利用率影响。根据现有功耗模型[23],PM_i 的总能耗定义为:

$$E_{PM_i}=E_{\text{cpu}}^{PM_i}+E_{\text{net}}^{PM_i}+E_{\text{io}}^{PM_i} \tag{7}$$

其中,$E_{\text{cpu}}^{PM_i}$,$E_{\text{net}}^{PM_i}$ 和 $E_{\text{io}}^{PM_i}$ 分别表示 CPU、网络带宽和磁盘 I/O 的能耗。具体地,CPU 能耗定义为:

$$E_{\text{cpu}}^{PM_i}=\begin{cases}\alpha_i\times Prh\times U_{\text{cpu}}^{PM_i}, & \text{if } U_{\text{io}}^{PM_i}\leqslant\lambda_i\\ \beta_i\times Prh+(1-\beta_i)\times Prh\times U_{\text{cpu}}^{PM_i}, & \text{if } U_{\text{io}}^{PM_i}>\lambda_i\end{cases} \tag{8}$$

其中,α_i 和 β_i 分别表示 PM_i 低 CPU 负载和高 CPU 负载水平系数;$Prh=P_{\max}-P_{\text{idle}}$ 表示物理机的最高与最低功率之差;λ_i 表示 $PM\,i$ 的负载阈值,当负载超过该阈值时,会从低负载 CPU 能耗模型切换至高负载 CPU 能耗模型。

网络带宽能耗定义为:

$$E_{\text{net}}^{PM_i}=\gamma_i\times Prh\times U_{\text{net}}^{PM_i} \tag{9}$$

其中,$U_{\text{net}}^{PM_i}$ 表示网络带宽利用率,γ_i 表示网络带宽利用率对能耗的影响系数。

磁盘 I/O 能耗定义为:

$$E_{\text{io}}^{PM_i}=\delta_i\times Prh\times U_{\text{io}}^{PM_i} \tag{10}$$

其中,$U_{\text{io}}^{PM_i}$ 表示磁盘 I/O 利用率,δ_i 表示磁盘 I/O 利用率对能耗的影响系数。

3.3　资源利用模型

在 Hadoop 集群中,物理机通常可为多个虚拟机提供计算、存储和网络资源。低效或不合理的资源利用造成了严重的资源浪费。对于 PM_i,其资源浪费可定义为:

$$W_{PM_i}=\begin{cases}0, & \text{if } U_{\text{cpu}}^{PM_i}+U_{\text{io}}^{PM_i}+U_{\text{net}}^{PM_i}=0\\ \dfrac{\sigma(U_{PM_i})+\epsilon}{\sum U_{PM_i}}, & \text{otherwise}\end{cases} \tag{11}$$

其中,U_{PM_i} 表示 PM_i 的资源利用率,包括 CPU、磁盘带宽和网络带宽利用率;ϵ 为一个正系数,用于避免标准差为零时出现不合理的极端情况;$\sigma(U_{PM_i})$ 表示 PM_i 上各项资源利用率的标准差,用于衡量资源的不平衡性,定义为:

$$\sigma(U_{PM_i})=\sqrt{\frac{\Delta PMi}{3}} \tag{12}$$

其中,$\Delta PM_i=(U_{\text{cpu}}^{PM_i}-\bar{U}_{PM_i})^2+(U_{\text{io}}^{PM_i}-\bar{U}_{PM_i})^2+(U_{\text{net}}^{PM_i}-\bar{U}_{PM_i})^2$。$\bar{U}_{PM_i}$ 表示各资源利用率的均值,定义为:

$$\bar{U}_{PM_i}=\frac{U_{\text{cpu}}^{PM_i}+U_{\text{io}}^{PM_i}+U_{\text{net}}^{PM_i}}{3} \tag{13}$$

因此,PM_i 上所有资源的利用率之和为:

$$\sum U_{PM_i}=U_{\text{cpu}}^{PM_i}+U_{\text{io}}^{PM_i}+U_{\text{net}}^{PM_i} \tag{14}$$

3.4　文件访问延迟模型

在 Hadoop 集群中,文件数据在物理机之间的传输需要经过多个交换机和物理链路。本文利用交换机延迟与物理链路延迟来评估虚拟机之间的文件访问延迟。具体地,交换机延迟 T_{switch} 指数据块通过交换机所需的处理时间。文件访问延迟为虚拟机所在物理机 PM_{dn} 到目标虚拟机所在物理机 PM_{vm} 之间所有交换机延迟和物理链路延迟之和,定义为:

$$L_{PM_{dn}\to PM_{vm}}=N\times T_{\text{switch}}+\sum_{k=1}^{M}T_{link_k} \tag{15}$$

其中,N 表示传输路径中的交换机数量,T_{link_k} 表示第 k 条物理链路的延迟。

当有 I 个数据块时,所有数据块的平均传输延迟 L_{avg} 可表示为:

$$L_{\text{avg}}=\frac{1}{I}\sum_{i=1}^{I}L_{PM_{dn_i}\to PM_{vm_j}} \tag{16}$$

3.5 问题定义

基于所提出的系统模型，本文的优化目标是最小化 Hadoop 集群的能耗、资源浪费和文件访问延迟的加权之和。具体地，该优化问题可形式化地定义为：

$$\begin{aligned}&\min(w_1Z_1+w_2Z_2+w_3Z_3)\\&\text{s.t. 式(1)，式(2)，式(6)}\end{aligned} \tag{17}$$

其中，$Z_1=\sum_{i=1}^{N}E_{PM_i}$，$Z_2=\sum_{i=1}^{N}W_{PM_i}$，$Z_3=\frac{1}{N}\sum_{i=1}^{N}L_{PM_{dn_i}\to PM_{vm_j}}$；$w_1$，$w_2$ 和 w_3 分别代表能耗、资源浪费和文件访问延迟 3 项优化目标的权重。

在此定义下，虚拟机的放置与数据块副本的分配需要综合考虑多重资源约束、负载动态变化以及文件访问延迟等因素。由于该问题属于 NP 难的多目标优化问题，传统方法在处理规模庞大、结构复杂的 Hadoop 集群时往往难以取得全局最优解。因此，第 4 章提出可变长度双染色体多目标优化(MO-VLD)方法，通过专门设计的编码结构、两阶段交叉与变异操作及参考点关联机制，实现对能耗、资源浪费和文件访问延迟的有效平衡。

4 MO-VLD 方法

为了有效求解上述多目标优化问题，本文提出了可变长度双染色体多目标优化(MO-VLD)方法，如图 2 所示。首先，基于系统模型中虚拟机与数据块的放置需求，设计了可变长度的染色体结构，由虚拟机染色体和块染色体两部分组成，以表示放置虚拟机和数据块副本的位置。接着，引入了两阶段交叉与变异操作以增强解空间探索的多样性。

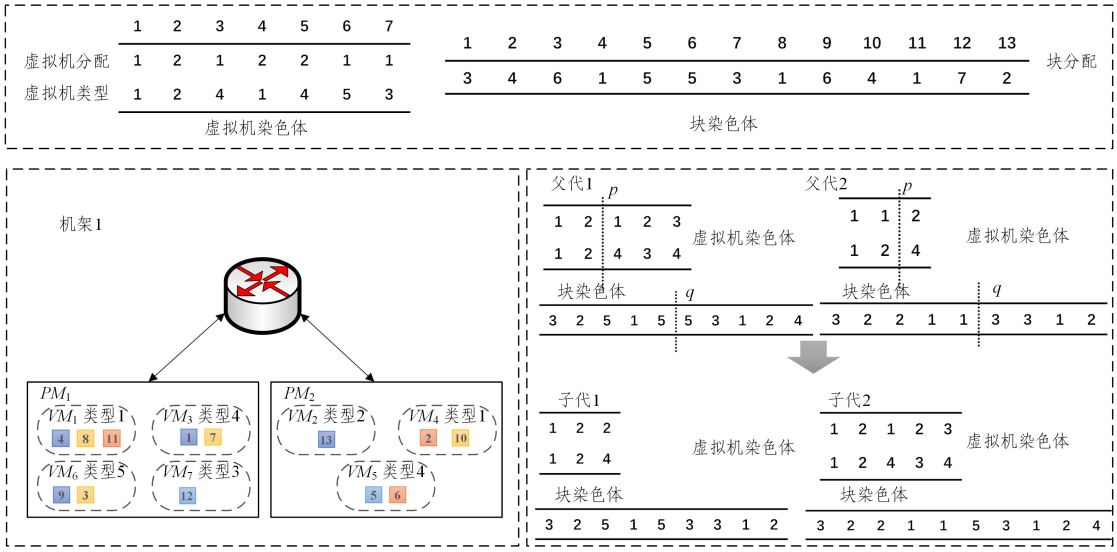


图 2 MO-VLD 方法的设计细节

Fig. 2 Design details of MO-VLD method

4.1 染色体设计

虚拟机染色体 C^{vm} 为二维数组，定义为：

$$C^{vm}=\begin{pmatrix}PM_1 & \cdots & PM_n\\ VMType_1 & \cdots & VMType_n\end{pmatrix} \tag{18}$$

其中，每列的第一行表示虚拟机被放置的物理机，每列的第二行表示虚拟机的类型，记为 $VMType_j\in\{1,2,3,4,5\}$。

块染色体 C^{block} 为一维数组，表示数据块副本的放置位置，定义为：

$$C^{\text{block}}=\{VM_1,VM_2,\cdots,VM_n\} \tag{19}$$

其中，每个元素表示数据块副本被放置的虚拟机。

在初始化阶段，C^{vm} 通过轮转方式将虚拟机放置到物理机上，并随机分配虚拟机类型。C^{block} 通过轮转和随机方式生成，确保数据块副本的分布符合 Hadoop 集群对副本数量的要求。上述初始化过程保证了解空间中的多样性，使得 MO-VLD 方法在运行阶段能够充分探索不同的策略。

4.2 交叉与变异操作

在进化过程中，采用交叉与变异操作生成新的解并优化现有解集。如图 2 所示，交叉操作分为以下两个阶段。

第一阶段对虚拟机染色体进行单点交叉。随机选择切割点 p，将不同父代虚拟机染色体的片段进行交换以生成子代虚拟机染色体。该过程可表示为：

$$\begin{aligned}C^{vm}_{child1}&=C^{vm}_1[0:p]\parallel C^{vm}_2[p+1:end]\\C^{vm}_{child2}&=C^{vm}_2[0:p]\parallel C^{vm}_1[p+1:end]\end{aligned} \tag{20}$$

第二阶段对块染色体进行单点交叉。随机选择切割点，将不同父代块染色体的片段进行交换以生成子代块染色体。该过程可表示为：

$$\begin{aligned}C^{\text{block}}_{child1}&=C^{\text{block}}_1[0:q]\parallel C^{\text{block}}_2[q+1:end]\\C^{\text{block}}_{child2}&=C^{\text{block}}_2[0:q]\parallel C^{\text{block}}_1[q+1:end]\end{aligned} \tag{21}$$

变异操作通过随机改变染色体中的基因来增加解的多样性。本文在虚拟机与块染色体上分别定义了变异操作。

1)对于虚拟机染色体，引入以下变异操作。(1)增加虚拟机：当系统负载上升时，增加新的虚拟机以扩展染色体长度。(2)减少虚拟机：当资源需求下降时，移除不必要的虚拟机以缩短染色体的长度。(3)分配虚拟机：通过重新分配虚拟机的位置来优化集群资源利用，以实现系统在动态环境下的高效运行。

2)对于块染色体，引入了块替换操作：通过改变数据块副本的位置来优化其存储位置，缩短数据传输延迟。具体地，当

一个数据块副本位于某个虚拟机时，块替换操作以 0.5 的概率将该副本重新分配至其他虚拟机。

4.3　参考点关联与归一化

参考点为预定义的目标向量，表示不同目标权重下的潜在解。具体地，本文将目标空间划分为多个区域，每个区域对应一个参考点。在每一代种群的进化过程中，当前解集将被投影到预先生成的参考点上，并根据二者之间的距离进行关联。接着，最近参考点将吸引解向其方向移动，进而优先选择与稀疏参考点相关联的解。基于参考点的选择算子维持了种群的多样性，并保证了解的均匀分布，从而避免解集中在某些目标区域而忽略其他维度的优化。

为了能够在相同尺度上对不同优化目标进行比较，引入了归一化操作。具体地，将个体在能耗、资源利用率和文件访问延迟上的性能指标转化为同一量级，从而消除不同优化目标之间的量级差异。该过程定义为：

$$r_i=\frac{1}{3}\frac{r_{1,i}}{\sqrt{\frac{\sum_{k=1}^{n} r_{1,k}}{n}}}+\frac{1}{3}\frac{r_{2,i}}{\sqrt{\frac{\sum_{k=1}^{n} r_{2,k}}{n}}}+\frac{1}{3}\frac{r_{3,i}}{\sqrt{\frac{\sum_{k=1}^{n} r_{3,k}}{n}}} \tag{22}$$

其中，r_i 表示第 i 个方案的加权目标值；$r_{1,i}$，$r_{2,i}$ 和 $r_{3,i}$ 分别表示能耗、资源浪费和文件访问延迟；n 表示帕累托解集的总数。

结合参考点关联机制与归一化操作，MO-VLD 方法能够实现多维目标的全局优化。

4.4　MO-VLD 方法的运行流程

MO-VLD 方法的关键步骤如算法 1 所示。首先，初始化最大时隙 T、最大迭代次数 X 和种群大小 N（第 1 行），并从谷歌集群的数据集中读取文件列表（第 2 行），以模拟真实云数据中心的负载需求。在每一个时隙内，首先初始化系统环境（第 4 行），包括物理机和虚拟机的配置信息。在每轮迭代过程中，利用轮转和随机化方法生成并初始化种群（第 7 行），以保证解集的多样性。接着，对虚拟机和块染色体进行单点交叉，并执行不同的变异操作和块替换操作，增加种群多样性（第 8－9 行）。随后，评估后代的适应度并与原始种群进行合并（第 10－11 行），以提高种群的质量。为了引导种群向帕累托最优前沿靠近，本文引入了预先生成的参考点，将当前解集投影到这些参考点上，进行非支配排序和拥挤距离的计算，进而根据两者关联度来选择下一代种群（第 12－13 行）。为保证方法的收敛性和全局搜索能力，在初始化阶段保存初始种群的适应度，并通过多次迭代提升种群质量。接着，从合并种群中选择最优解作为下一代种群，并保存当前的帕累托前沿（第 14－15 行）。最后，通过归一化，保证不同优化目标之间的平衡性并选取最佳的虚拟机放置策略（第 17－18 行）。

算法 1　MO-VLD 方法

输入：复制因子 replication_factor，数据块大小 block_size，机架数量 rack_num，物理机配置 pm，虚拟机配置 vm，优化目标权重 energy_weight，res_weight，latency_weight

输出：最佳的虚拟机放置策略

1. 初始化最大时隙 T、最大迭代次数 X、种群大小 N；
2. 从谷歌集群数据集中读取文件列表；
3. FOR t=1，2，…，T DO
4.　　初始化系统环境：env. con_system( )；
5.　　FOR x=1，2，…，X DO
6.　　初始化种群：InitializePopulation( )；
7.　　对虚拟机和块染色体进行单点交叉：SinglePointCrossover( )；
8.　　执行变异操作：MutationOperation( )；
9.　　评估后代的适应度：EvaluateFitness( )；
10.　　与原始种群合并：PopulationMerge( )；
11.　　进行非支配排序：NonDomSort( )；
12.　　计算拥挤距离：crowdedComparisonOrder( )；
13.　　合并种群中选择下一代：Selection( )；
14.　　保存当前的帕累托前沿；
15.　　END FOR
16.　　保存种群结果；
17.　　归一化之后选取最佳的虚拟机放置策略；
18. END FOR

5　实验评估

5.1　实验设置

实验在配备了 Intel(R) Xeon(R) Silver 4208 CPU 和 NVIDIA Geforce GTX 3090 GPU 的工作站上开展，其 CPU 频率为 2.1 GHz、内存为 32 GB。实验基于 Python 3.9，搭建了大规模 Hadoop 集群环境，其配置参考了典型的云数据中心[15]，并使用了谷歌集群数据集[24]。该数据集记录了 7 小时内从 Borg 单元获取的运行时数据，反映了云数据中心中任务调度和资源利用情况。根据每个时隙内任务的到达情况来初始化环境，以模拟具有动态变化负载的场景。物理机和虚拟机的具体设置如表 1 和表 2 所列，文件系统的设置如表 3 所列。此外，将 MO-VLD 方法与以下 5 种方法进行了对比。

1）第 2 代非支配排序遗传算法（Non-dominated Sorting Genetic Algorithm II，NSGA-II）[16]：通过非支配排序和拥挤度进行子代的选择。

2）多目标粒子群优化算法（Multi-Objective Particle Swarm Optimization，MOPSO）[7]：利用粒子群的全局搜索能力优化多个目标。

3）多目标模拟退火算法（Multi-Objective Simulated Annealing，MOSA）[8]：通过温度控制的逐步退火过程找到近似最优解。

4）最佳适应下降算法（Best-Fit Decreasing，BFD）：优先将虚拟机放置到最满足其资源需求的物理机中。

5）首次适应下降算法（First-Fit Decreasing，FFD）：根据顺序优先将虚拟机放置在首个能满足其资源需求的物理机中。

在实验中，所有方法的种群大小均设置为 200，迭代次数为 300 次，采用轮转和随机策略进行种群初始化。MO-VLD 和 NSGA-II 方法采用两阶段交叉策略进行交叉操作。MO-VLD 方法通过虚拟机扩展、减少、分配以及块替换来实现变异操作，NSGA-II 方法仅通过块替换作为变异操作。变异概率均为 0.25。MOPSO 方法的惯性权重、认知权重和社会权重分别设置为 0.5，1.5 和 1.5，此参数配置基于 Clerc 和 Kennedy 提出的惯性权重与学习因子的理论基础，将惯性权重稍微降低到 0.5，平衡对全局与局部搜索的关注度，以便在收敛到较优解的同时保持一定的探索能力。MOSA 方法的初始

温度和降温速度分别为 100 和 0.99，以在算法早期提供充分的随机扰动并在后期稳步降低温度，从而在搜索范围与收敛速度之间取得平衡。

表 1 物理机的设置[23]

Table 1 Settings of PMs[23]

参数	PM_0	PM_1
λ_i	0.12	0.12
α_i	5.29	4.33
β_i	0.68	0.47
γ_i	0.05	0.05
δ_i	0.1	0.1
U^{cpu}_{hypmi}/Cores	24	12
U^{io}_{hypmi}/(MB/s)	17800	15000
U^{net}_{hypmi}/(MB/s)	76800	38400

表 2 虚拟机的设置[25]

Table 2 Settings of VMs[25]

虚拟机类型	内核数	磁盘带宽/(MB/s)	网络带宽/(MB/s)
c3.xlarge	4	250	100
c3.2xlarge	8	320	240
c3.4xlarge	16	400	200
m3.xlarge	4	320	100
m3.2xlarge	8	400	200

表 3 文件系统的设置[16]

Table 3 Settings of file system[16]

参数	描述	值
$fbSize$/MB	块大小	64
$fRfu$	复制因子	3
$fSize$/MB	文件大小	[1600,1623,1646,1671,1696,1723,1750,1779,1809,1839,1872,1905,1941,1977,2016,2056,2099,2143,2190,2239,2292,2347,2406,2468,2535,2606,2681,2763,2851,2946,6049,3161,3283,3418,3567,3733,3918,4128,4367,4643,4965,5347,5810,6382,7113,8087,9462,15412,25101]
$fRate$/ms^{-1}	访问速率	[128.21,78.97,58.46,48.21,41.03,35.90,31.79,29.74,26.67,24.62,23.59,21.24,20.51,19.49,18.46,17.44,17.44,16.41,15.38,15.38,14.36,14.36,13.33,13.33,13.33,12.31,12.31,12.31,11.28,11.28,11.28,10.26,10.26,10.26,10.26,9.23,9.23,9.23,9.23,9.23,9.23,8.21,8.21,8.21,8.21,8.21,8.21,8.21,7.18]

5.2 结果分析

5.2.1 收敛性与帕累托解集分布对比

首先，对比了不同方法的收敛性与帕累托解集分布。设置文件数量为 200，物理机数量为 40，120 和 200，进而比较不同方法的收敛性与解的质量。图 3—图 5 分别对比了在 40，120 和 200 个物理机场景下不同方法的收敛性。可以看出，随着迭代次数的增加，所有方法均逐渐收敛。MO-VLD 方法能够在较少的迭代次数内快速收敛，并且其最终优化效果优于其他方法。这是因为 MO-VLD 方法采用了双染色体结构和两阶段交叉变异操作，增加了解的多样性，避免了方法陷入局部最优。同时，MO-VLD 方法引入了参考点关联机制，能够更好地平衡多个优化目标，确保各优化目标均能得到有效优化。相比之下，NSGA-II 方法虽然在迭代早期具有较快的收敛速度，但其交叉变异操作设计过于简单，导致其后期优化效果略逊一筹。MOPSO 方法易陷入局部最优，而 MOSA 方法虽然在早期迭代时易取得较好的解，但退火过程导致其收敛缓慢。具体而言，如图 3 所示，当物理机数量为 40 时，相比 NSGA-II，MOPSO 和 MOSA 方法，MO-VLD 方法在能耗方面分别降低了约 3.12%，7.9% 和 4.3%，在资源利用率方面分别提升了约 1.4%，7.6% 和 9%，在文件访问延迟方面分别降低了约 0.7%，17.2% 和 22.2%。如图 4 所示，当物理机数量为 120 时，相比 NSGA-II，MOPSO 和 MOSA 方法，MO-VLD 方法在能耗方面分别降低了约 4.47%，6.70% 和 9.3%，在资源利用率方面分别提升了约 5.14%，5.25% 和 13.04%，在文件访问延迟方面分别降低了约 1.37%，4.17% 和 6.91%。如图 5 所示，当物理机数量为 200 时，相比 NSGA-II，MOPSO 和 MOSA 方法，MO-VLD 方法在能耗方面分别降低了约 1.0%，1.8% 和 5.8%，在资源利用率方面分别提升了约 0.9%，5.9% 和 7.4%，在文件访问延迟方面分别降低了约 1.3%，8.5% 和 10.3%。

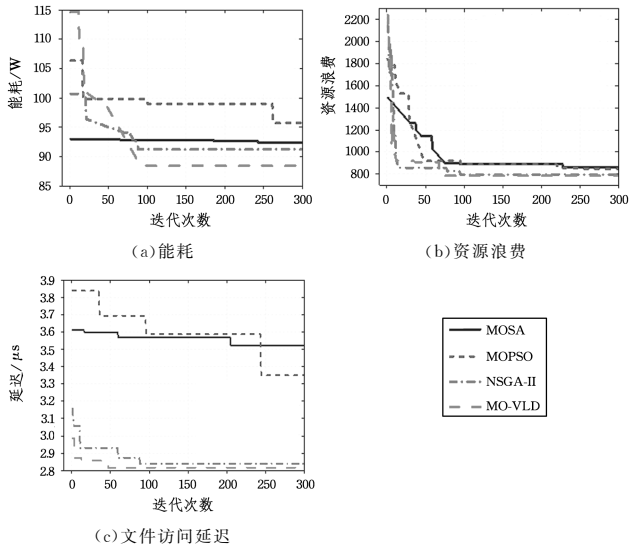


图 3 40 个物理机场景下不同方法的收敛性对比

Fig. 3 Convergence comparison of different methods under the scenario with 40 PMs

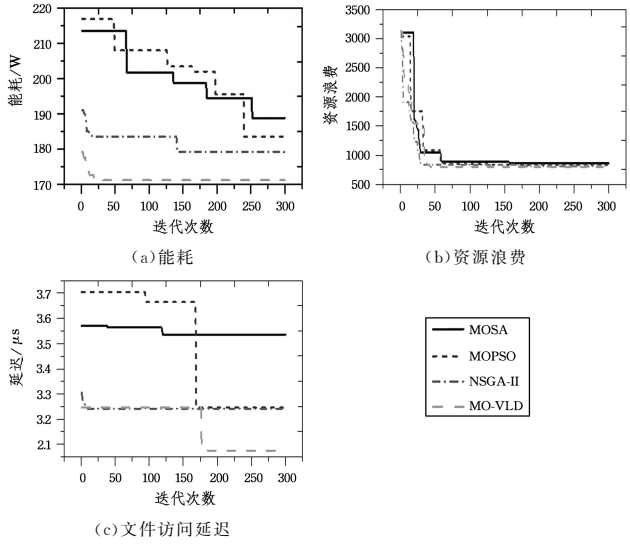
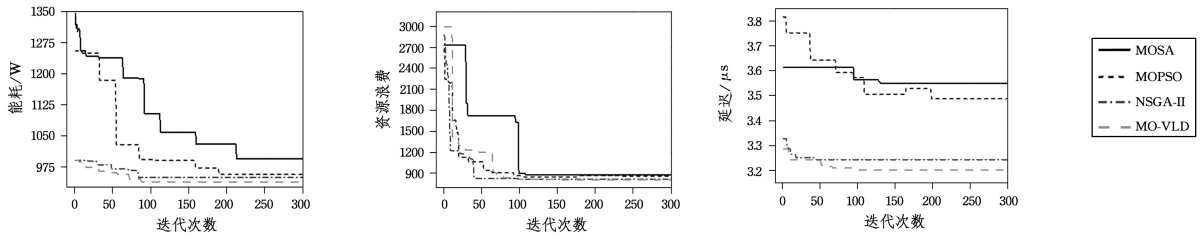


图 4 120 个物理机场景下不同方法的收敛性对比

Fig. 4 Convergence comparison of different methods under the scenario with 120 PMs

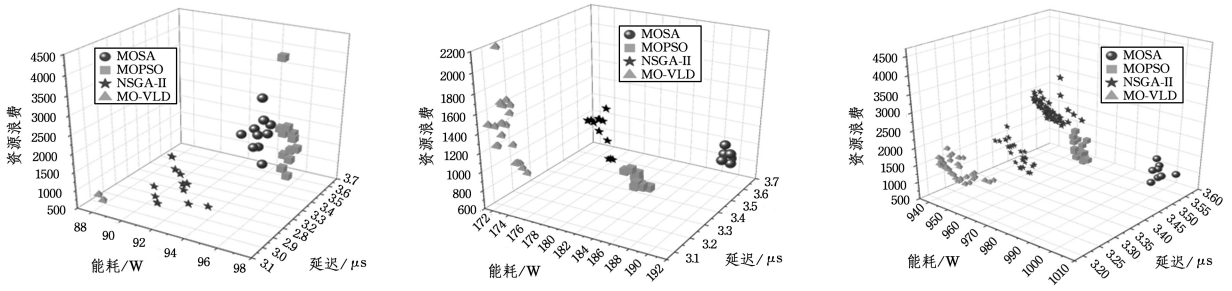


(a)能耗　(b)资源浪费　(c)文件访问延迟

图 5　200 个物理机场景下不同方法的收敛性对比

Fig. 5　Convergence comparison of different methods under the scenario with 200 PMs

图 6 展示了不同方法的帕累托解集分布。在这几种方法中，MOSA，MOPSO，NSGA-II 和本文 MO-VLD 方法的性能依次提高。相比于其他 3 种方法，MO-VLD 方法的帕累托解集更加趋向于三维坐标轴的原点，这说明 MO-VLD 方法得到的解的质量更高。这是因为 MO-VLD 方法引入了不同的交叉和变异操作，所以其全局搜索能力优于其他方法。随着物理机数量的增加，帕累托解集的数量也在增加，这表明随着场景复杂程度增加，潜在可行解的数量也会增加。



(a)40 个物理机　(b)120 个物理机　(c)200 个物理机

图 6　不同方法的帕累托解集分布

Fig. 6　Distribution of Pareto solution sets of different methods

5.2.2　不同物理机数量场景下的性能对比

进一步地，本文对比了不同方法在不同物理机数量场景下的性能。物理机数量设置为 40，80，120，160 和 200，虚拟机数量为物理机的两倍。当物理机数量低于 100 时，机架数量为 5；否则，机架数量为 8。通过上述设置以模拟大规模集群的配置。本文对能耗、资源浪费和文件访问延迟这 3 个性能指标进行了归一化与加权处理，进而比较不同方法的性能。如图 7 所示，MO-VLD 方法的加权目标值是最低的，这是因为 MO-VLD 方法采用了双染色体结构，允许虚拟机和数据块副本放置的同步优化，大大减少了跨机架的数据传输和物理机资源的不均衡使用。同时，MO-VLD 方法中的两阶段交叉与变异操作也增强了其对解空间的探索能力。因此，MO-VLD 方法在应对多目标优化问题时表现出色，无论物理机数量如何变化，其始终能保持良好且稳定的优化效果。NSGA-II 方法的加权目标值仅约为 FFD 和 BFD 方法的一半，比 MOPSO 和 MOSA 方法低约 0.05。NSGA-II 方法虽然在各场景下的表现也较为稳定，但其交叉和变异操作相对简单，在大规模集群场景中难以有效平衡多种优化目标，导致其性能稍逊于 MO-VLD 方法，特别是在物理机增加至 200 的场景中。MOPSO 和 MOSA 方法的表现始终不如 MO-VLD 方法。这是因为 MOPSO 方法在粒子群搜索过程中易陷入局部最优，难以在多目标优化问题中取得全局均衡。MOSA 方法尽管具备一定的全局搜索能力，但其退火过程导致运行开销过大，在面对大规模集群时表现不佳。FFD 和 BFD 属于启发式算法，其在面对大规模集群时严重缺乏全局优化能力，因此优化效果最差。

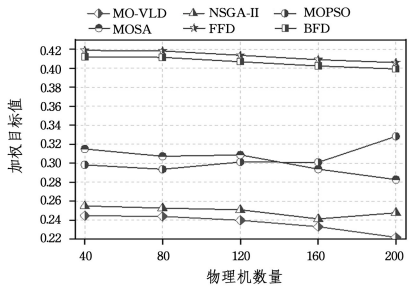


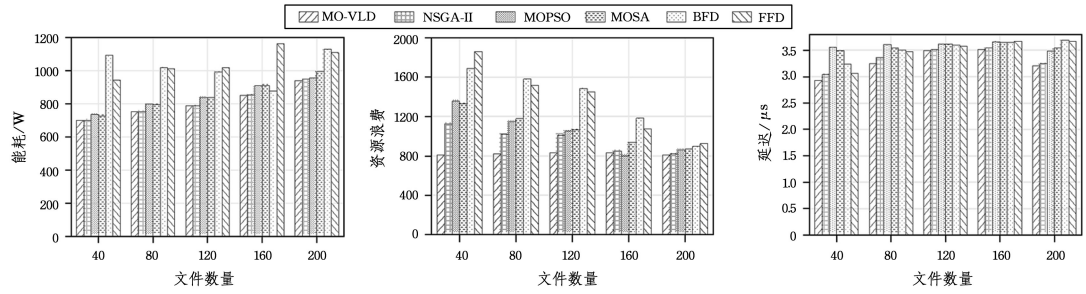
图 7　不同物理机数量场景下不同方法的性能对比

Fig. 7　Performance comparison of different methods under scenarios with various numbers of PMs

5.2.3　不同文件数量对性能的影响

最后评估了不同文件数量对不同方法性能的影响。物理机数量设置为 200，文件数量设置为 40，80，120，160 和 200。如图 8 所示，MO-VLD 方法在不同文件数量场景下均能达到最优的性能表现。在能耗方面，MO-VLD 在所有文件数量场景中的能耗均低于其他方法，并且随着文件数量的增加，均能保持较低的能耗。这是因为 MO-VLD 方法中的双染色体结构能够实现虚拟机和数据块位置最佳放置，避免了资源浪费或过度使用。相比 NSGA-II，MOPSO，MOSA，FFD 和 BFD 方法，MO-VLD 方法在不同场景下平均能取得约 0.4%，5.0%，5.6%，23.6% 和 26.2% 的性能提升。随着文件数量的增加，所有方法的资源浪费程度均呈下降趋势。相比于其他方法，MO-VLD 方法在不同文件数量场景中皆能保持稳定

的低资源浪费。这是因为 MO-VLD 方法采用了动态资源调度和块替换机制，可在资源需求变化时及时调整。相比于 NSGA-II，MOPSO，MOSA，FFD 和 BFD 方法，MO-VLD 方法在不同场景下平均取得了约 16.2%，26.0%，28.9%，35.1% 和 38.4% 的性能提升。在文件访问延迟方面，虽然不同方法性能相差不大，但 MO-VLD 方法仍有着不错的表现。这说明 MO-VLD 方法能够有效缩短数据传输时间，提高集群的文件访问效率。具体而言，与 NSGA-II，MOPSO，MOSA，FFD 和 BFD 方法相比，MO-VLD 方法在不同场景下平均取得了约 1.8%，8.9%，8.5%，7.5% 和 6.2% 的性能提升。



(a)能耗 (b)资源浪费 (c)文件访问延迟

图 8 不同文件数量对不同方法性能的影响

Fig. 8 Impact of various numbers of files on the performance of different methods

结束语 本文针对大规模 Hadoop 集群中的虚拟机放置问题，设计了基于可变长度双染色体的多目标优化(MO-VLD)方法。首先，将虚拟机放置问题形式化为一个多目标优化问题。接着，通过两阶段单点交叉变异操作和块替换机制增加解空间的探索多样性。最后，利用参考点关联和非支配排序选择最优解。基于谷歌集群真实运行数据集，通过大量实验验证了 MO-VLD 方法的有效性。在不同物理机与文件数量场景下，相比于其他基准方法，MO-VLD 方法在能耗、资源利用率、文件访问延迟方面均展现出了更加优越的性能。

然而，本文的研究仍存在若干局限性。首先，MO-VLD 方法在虚拟机和文件副本的实时迁移过程中未充分考虑迁移成本，这可能使其在高动态负载环境下造成额外的资源消耗和系统开销。此外，MO-VLD 方法的算法实现和计算效率尚有提升空间。在面对突发高负载或资源需求急剧变化时，MO-VLD 的计算效率和资源消耗可能成为瓶颈，适应能力需要进一步优化。

未来的研究将致力于降低 MO-VLD 方法的实时迁移成本，探讨更高效的算法设计和并行计算技术，同时进一步完善虚拟机放置策略，以提升其在大规模集群中的应用可行性。通过结合更多样化的数据集和实际应用案例，MO-VLD 方法的普适性和有效性可以得到更有效的验证，以推动其在实际云数据中心中的广泛应用。

参 考 文 献

[1] MIAO C, ZHONG Z, XIAO Y, et al. MegaTE: Extending WAN Traffic Engineering to Millions of Endpoints in Virtualized Cloud[C]// Proceedings of the ACM SIGCOMM 2024 Conference. 2024: 103-116.

[2] WEI C, LI X, YANG Y, et al. Achelous: Enabling Programmability, Elasticity, and Reliability in Hyperscale Cloud Networks [C]// Proceedings of the ACM SIGCOMM 2023 Conference. 2023: 769-782.

[3] TAKAMORI D. HDFS Users Guide. [EB/OL](2023-06-18). https://hadoop. apache. org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide. html.

[4] TAKAMORI D. MapReduce Tutorial. [EB/OL](2022-05-18). https://hadoop. apache. org/docs/r1. 2. 1/mapred _ tutorial. html.

[5] CHEN Q, HUANG W, HUANG Y. The Learnable Model-Based Genetic Algorithm for the IP Mapping Problem[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, 42(7): 2350-2363.

[6] AYERDI J, TERRAGNI V, JAHANGIROVA G, et al. Automatically Generating Metamorphic Relations via Genetic Programming[J]. arXiv: 2312. 15302, 2023.

[7] BRAIKI K, YOUSSEF H. Multi-objective virtual machine placement algorithm based on particle swarm optimization[C]// 2018 14th International Wireless Communications & Mobile Computing Conference(IWCMC). IEEE, 2018: 279-284.

[8] BHATT C, SINGHAL S. Hybrid Metaheuristic Technique for Optimization of Virtual Machine Placement in Cloud[J]. International Journal of Fuzzy Logic and Intelligent Systems, 2023, 23(3): 353-364.

[9] SRIVASTAVA A, KUMAR N. Virtual Machine Allocation Using Genetic-Based Algorithm in Cloud Infrastructure[C]// Proceedings of Second International Conference on Computational Electronics for Wireless Communications: ICCWC 2022. Singapore: Springer, 2023: 273-282.

[10] YARAHMADI A, MOMTAZPOUR M. VM placement in accelerator-equipped data centers using variable-length modified genetic algorithm[C]// 2021 29th Iranian Conference on Electrical Engineering(ICEE). IEEE, 2021: 562-567.

[11] SWAIN S R, PARASHAR A, SINGH A K, et al. A Multi-objective Virtual Machine Placement Optimization in Sustainable Cloud Environment [C] // International Conference on Deep Learning, Artificial Intelligence and Robotics. Cham: Springer, 2023: 415-426.

[12] TANG M, PAN S. A Hybrid Genetic Algorithm for the Energy-Efficient Virtual Machine Placement Problem in Data Centers [J]. Neural Processing Letters, 2015, 41(2): 211-221.

[13] GOPU A，THIRUGNANASAMBANDAM K，ALGHAMDI A S，et al. Energy-efficient virtual machine placement in distributed cloud using NSGA-III algorithm[J]. Journal of Cloud Computing，2023，12(1)：124.

[14] CONEJERO J，CAMINERO B，CARRIÓN C. Analysing Hadoop performance in a multi-user IaaS Cloud[C]// 2014 International Conference on High Performance Computing & Simulation(HPCS). IEEE，2014：399-406.

[15] HEDAYATI S，MALEKI N，OLSSON T，et al. MapReduce scheduling algorithms in Hadoop：a systematic study[J]. Journal of Cloud Computing，2023，12(1)：143.

[16] GUERRERO C，LERA I，BERMEJO B，et al. Multi-Objective Optimization for Virtual Machine Allocation and Replica Placement in Virtualized Hadoop[J]. IEEE Transactions on Parallel and Distributed Systems，2018，29(11)：2568-2581.

[17] MARQUEZ J，MONDRAGON O H，GONZALEZ J D. An Intelligent Approach to Resource Allocation on Heterogeneous Cloud Infrastructures[J]. Applied Sciences，2021，11(21)：9940.

[18] LI Y，HEI X. Performance optimization of computing task scheduling based on the Hadoop big data platform[J]. Neural Computing and Applications，2022，37：8181-8192.

[19] GHAZALI R，ADABI S，DOWN D G，et al. A classification of Hadoop job schedulers based on performance optimization approaches[J]. Cluster Computing，2021，24(4)：3381-3403.

[20] ZHANG Y，ZHANG X. Minimizing Data Access Latencies via Virtual Machine Placement Method in Datacenter[C]// 2017 14th International Symposium on Pervasive Systems，Algorithms and Networks & 2017 11th International Conference on Frontier of Computer Science and Technology & 2017 Third International Symposium of Creative Computing(ISPAN-FCST-ISCC). IEEE，2017：197-202.

[21] NAIK S，KALRA M. Big Data Processing with Balanced Resource Utilization[C]// 5th International Conference on Next Generation Computing Technologies. 2020.

[22] MIRIAM A J，SAMINATHAN R，CHAKARAVARTHI S. Non-dominated Sorting Genetic Algorithm(NSGA-III) for effective resource allocation in cloud[J]. Evolutionary Intelligence，2021，14：759-765.

[23] DE MAIO V，KECSKEMETI G，PRODAN R. An improved model for live migration in data centre simulators[C]// Proceedings of the 9th International Conference on Utility and Cloud Computing. 2016：108-117.

[24] Johnwilkes. Google Cluster data[EB/OL]. (2020-04-02). https://github. com/google/cluster-data/blob/master/TraceVersion1. md.

[25] Amazon. Amazon EC2instance types [EB/OL]. (2024-09-25). https://aws. amazon. com/cn/ec2/instance-types/.

WEN Jia，born in 2000，postgraduate，is a member of CCF(No. P7488G). Her main research interests include cloud/edge computing and virtual machine placement.

CHEN Zheyi，born in 1991，Ph. D，professor，Ph.D supervisor，is a member of CCF(No. 41902M). His main research interests include cloud/edge computing，resource optimization and machine learning.

（责任编辑：何杨）