2021 年 1 月 第 1 期
Vol. 42 No. 1 2021

面向移动云计算的自适应虚拟机调度算法

韦传讲,庄 毅

(南京航空航天大学 计算机科学与技术学院,南京 211106)
E-mail:zy16@nuaa.edu.cn

摘 要: 随着移动云计算的快速发展和应用普及,如何对移动云中心资源进行有效管理同时又降低能耗、确保资源高可用是目前移动云计算数据中心的热点问题之一. 本文从 CPU、内存、网络带宽和磁盘四个维度,建立了基于多目标优化的虚拟机调度模型 VMSM-EUN(Virtual Machine Scheduling Model based on Energy consumption,Utility and minimum Number of servers),将最小化数据中心能耗、最大化数据中心效用以及最小化服务器数量作为调度目标. 设计了基于改进粒子群的自适应参数调整的虚拟机调度算法 VMSA-IPSO(Virtual Machine Scheduling Algorithm based on Improved Particle Swarm Optimization)来求解该模型. 最后通过仿真实验验证了本文提出的调度算法的可行性与有效性. 对比实验结果表明,本文设计的基于改进粒子群的自适应虚拟机调度算法在进行虚拟机调度时,能在降低能耗的同时提高数据中心效用.

关 键 词: 移动云数据中心; 能源消耗; 虚拟机调度; 自适应参数调整; 粒子群优化
中图分类号: TP311 **文献标识码:** A **文 章 编 号:** 1000-1220(2021)01-0096-09

Self-adaptive Virtual Machine Scheduling Algorithm for Mobile Cloud Computing

WEI Chuan-jiang,ZHUANG Yi
(Department of Computer Science and Technology,Nanjing University of Aeronautics and Astronautics,Nanjing 211106,China)

Abstract: With the rapid development of mobile cloud computing and the popularity of applications,How to effectively manage mobile cloud center resources while reducing energy consumption and ensuring high resource availability is one of the hot issues in mobile cloud computing data centers. In this paper,focusing on four dimensions of CPU,memory,network bandwidth,and disk,we establish a virtual machine scheduling model VMSM-EUN based on multi-objective optimization which can minimize the data center energy consumption,maximize data center utility,and minimize the number of servers. We propose a virtual machine scheduling algorithm VMSA-IPSO based on improved particle swarm optimization to solve the model. The improvement includes adaptive parameter adjustment. Finally,the effectiveness of the proposed algorithm is verified by simulation experiments. The experimental results show that the adaptive virtual machine scheduling algorithm can improve the efficiency of data center while reducing energy consumption.

Key words: mobile cloud data center; energy consumption; VM scheduling; self-adaptive parameter adjustment; particle swarm optimization

1 引 言

随着无线网络的快速发展,促使了诸如手机、笔记本和平板电脑之类的智能设备的发明,这样的联网技术和设备使得用户可以移动,并可以随时随地访问资源. 因此,移动性已成为现代互联网世界的主要特征. 根据中国互联网信息中心发布的报告,截止 2018 年底,我国网民数量大约为 8.29 亿,其中大概有 98.6% 的网民是通过手机访问互联网[1]. 随着互联网技术的不断发展,应用服务的种类变得多样化,使得人们对移动设备的计算性能的需求也越来越高[1].

移动设备功能和性能需求的上升需要不断突破本身的资源瓶颈,才能满足人们的需求. 然而,硬件的提升需要新材料新技术的突破. 目前移动设备存在的难以解决的问题,包括移动设备的存储有限,电池续航能力有限,设备的处理能力有限等,都需要新材料技术的突破,才能够使移动设备的性能达到质的飞跃,而这些突破无法在短期实现[2]. 而云计算技术的出现,使得移动设备固有的缺点得以弥补. 云平台上有近乎无限的计算资源和存储资源,能够满足移动终端的各种高性能计算,并提供给移动终端最优的云服务.

移动云计算技术结合了移动互联网技术与传统云计算技术,改善了移动设备固有的缺点. 云平台上的计算资源和存储资源被认为几乎是无限的,可以满足移动设备的多种计算需求,使得移动设备获得更好的云服务[3]. 移动云计算能够支持多种移动设备执行丰富的移动应用,可以为用户带来丰富的产品体验. 同时,移动云计算也为移动网络运营商和云服务提供商带来了商业机会[4]. 移动云计算相较于传统云计算而言,它是一种先进的移动计算技术,其利用各种云和网络技术的统一弹性资源获得不受限的功能、存储和移动性,它弱化了

[1] http://www.cnnic.net.cn/hlwfzyj/hlwxzbg/hlwtjbg/201902/t20190228_70645.htm.

收稿日期:2020-01-05 收修改稿日期:2020-04-28 基金项目:国家自然科学基金项目(61572253)资助;航空科学基金项目(2016ZC52030)资助. 作者简介:韦传讲,男,1995 年生,硕士研究生,CCF 会员,研究方向为云计算、机器学习;庄 毅,女,1956 年生,教授,博士生导师,研究方向为网络安全、分布计算.

移动终端硬件设备的限制,通过以太网或互联网渠道随时随地地为大量的移动终端提供各种定制的云服务,而不用管其基于的终端平台[5]. 此外,移动云计算还继承了传统云计算的一些优势,如可伸缩性、支持多用户以及安全服务[6].

移动云计算的日益普及和云中心数量与规模的不断扩大,使得能源消耗成为云中心的最大的运营成本. 关于能耗的数字统计随处可见,根据《中国数据中心白皮书(2018 年)》[2]显示,截止 2017 年底全球数据中心共计 44.4 万个,大约安装了 493.3 万架机架,部署至少 5500 万台服务器,预计 2020 年会部署安装更多的机架和服务器,这些服务器消耗着越来越多的能源. 因此世界上很多国家或组织已经在关注云数据中心高能耗的问题,并付诸行动. 移动云计算在满足云用户不断增长应用需求的同时,需要对工作负载和资源配置进行快速智能化和动态管理,从而实现能耗最小和效益最大化的运行模式. 因此,很有必要研究相关算法以支持对移动云中心的资源管理,降低开销,提高效率和可靠性.

本文针对移动云中心高能耗问题,从移动云中心资源调度的角度展开研究,设计目标函数,并建立虚拟机(Virtual Machine,VM)调度模型,最后提出 VM 调度算法以对模型进行求解. 本文的主要研究工作及贡献如下:

1) 从 CPU、内存、网络带宽和磁盘 4 个维度,将最小化数据中心能耗、最大化数据中心效用以及最小化服务器数量作为目标,建立了基于多目标优化的 VM 调度模型 VMSM-EUN. 该模型具有能耗低、效用高和调度结果优等特点.

2) 根据粒子群进化的不同阶段,分别设计了适用于加速因子与惯性因子的动态自适应调整策略,可加速粒子群的求解过程并获得更优解.

3) 为验证 VM 调度模型 VMSM-EUN 的有效性,设计了基于改进粒子群的自适应参数调整的 VM 调度算法 VMSA-IPSO 来求解该模型. 并通过仿真实验验证了本文算法与模型的有效性,并通过对比实验验证了算法的优越性.

2　相关研究

在移动云计算的众多研究中,通过对虚拟资源的合理调度来降低数据中心的能耗一直是一个研究热点. 国内外研究学者从多个方向提出了提高云数据中心的资源利用率和能源效率的方案,包括新颖的资源调度方案、数据中心服务器散热和控温方案、服务器调压变频方案以及高效负载均衡器等. 其中资源调度方案对数据中心的能耗影响尤为突出,但设计更快速、更节能的资源调度算法被认为是一个挑战.

目前,许多解决方案采用的分配方法是通过在任意给定时间将单个 VM 分配给主机的方式,来解决 VM 分配问题. 这种方法独立地规定了 VM 的资源需求,以确保每个主机具有足够的容量来执行工作负载. 但是,这种方法会降低资源利用效率. 此外,许多已有的 VM 部署方案还采用了一种根据当前的 CPU 资源需求优化资源使用的部署策略. 但应用程序的工作负载总是会随着时间的推移而波动,这对 VM 的动态调度是一个重大的挑战. 最近的一些研究表明,通过分析、预测 CPU 资源和云网络流量也能为云中心有效管理资源提供帮助,确保资源高效合理的分配. 为了解决 VM 调度问题,已有许多专家学者展开了研究,主要包括以降低能耗为目标和以收益最大化为目标.

以降低能耗为目标,对移动云数据中心的资源进行管理和调度,提高云中心资源的利用率和能源效率. Li xiang 等人将数据中心的能耗分为计算能耗和冷却能耗,他们精心设计了热能耗模型来分析数据中心的气流和服务器 CPU 的温度分布,提出了一种能够最大限度地降低云数据中心总能耗的整体 VM 调度算法[7]. 文献[8]提出了一种通过对服务器的状态进行管理来减少能耗的方法. 将空闲服务器节电状态与输入作业负载进行映射,考虑了资源动态变化情况、工作服务器不同负载时的功耗和空闲服务器休眠状态及转换时的功耗等情况,设计了启发式调度器来对 VM 进行调度,以使得 VM 映射中功耗增量最小. 文献[9]提出一种能量感知 VM 调度方法,该方法考虑了数据中心的物理主机的异构带宽,由功率感知算法和带宽供应算法组成. Yibin Li 等人在 DVS 技术的基础上,提出了一种新的能量感知移动云动态资源调度算法[10],在满足严格的时间约束和应用的概率约束的前提下,可减少移动设备的总能耗. 文献[11]提出了一个混合云框架 H^2 和集成资源管理器,该框架和管理器能够管理具有多种沙箱化和虚拟化技术的云基础设施,可有效地平衡工作负载的能源、性能和成本效益. Jang 等人提出了一种新的虚拟 CPU 调度方案,以提高云应用的 I/O 性能并减少能耗. 该方案设计了一个评估函数,该评估函数通过观察每个虚拟 CPU 的的资源消耗量来评估 VM 的任务特性. 基于评价函数,该调度方案自适应地控制 VM 在处理 I/O 请求时的优先级,以实现公平分配. 实验结果表明,该方案提高了 VM 的响应速度和 I/O 带宽,并降低能耗[12]. Ting Yang 等人提出了一种新颖的工作感知 VM 布局[13],以减少数据中心的能耗,并尽可能多地满足网络 QoS 要求. 并通过实验验证了其方案可在节省能源消耗的同时减小通信延迟. 文献[14]提出了一种 VM 整合的自适应节能调度算法,该算法的主要思想是最大限度的利用服务器,将数据中心的 VM 部署在尽可能少的服务器上,以最小化能耗. 该算法通过自组织和概率论方法来阻止过载服务器接受 VM 请求,以确保良好的 QoS,并使用在线迁移来确保迁移过程的顺利进行. 实验结果表明,该算法能降低数据中心能耗实现节能目标[14].

以收益最大化为目标,主要有基于机器学习算法、基于拍卖机制和博弈论的方法. 文献[15]提出了一种考虑云用户和云服务提供商的双激励 VM 调度模型,该模型通过最大化 VM 请求的成功率和最小化组合成本实现对云用户的激励,并通过最小化利润的公平性偏差实现对云服务提供商的激励. 上述的策略能使获得更高的用户满意度,但是容易造成资源的浪费. Sharrukh Zaman 等人针对现有云资源分配策略无法保证有效的分配和云提供商收入最大化的问题,设计了一种基于拍卖的动态 VM 分配机制,该机制在进行 VM 配置决策时考虑了用户对 VM 的需求. 实验表明所提出的机制可以提高资源利用率和分配效率,并为云提供商带来更高收益[16]. Liu Xing 等人提出了一种基于 Stackellberg 博弈模型的 VM 定价和分配方案,该方案考虑服务阻塞对用户效用的影响,通过模型分析推导出最优定价的闭式表达式,并通过粒

[2] http://www.caict.ac.cn/kxyj/qwfb/bps/201810/t20181016_186900.htm.

子群算法对 VM 静态和动态调度进行最优定价搜索. 实验结果证明了该方案可以通过定价策略控制网络阻塞,有效提高移动用户和云提供商的实用性[17]. 文献[18]对移动云环境下应用任务需求进行分析,设计了基于 Stackelberg 博弈的动态计算资源调度算法和博弈模型,仿真实验表明该算法和模型能保证移动终端的体验质量并最大化终端效用[18].

综上所述,目前已有的云中心的 VM 调度方法虽然在一定程度上减少了云中心的能耗,但是仍然存在不足,一方面,当 VM 调度的数量很大时,现有的算法不能满足调度的性能需求. 另一方面,上述的 VM 调度算法在设计时考虑问题不够全面,如仅考虑 CPU 能耗而不考虑其他部件的能耗问题. 因此,本文在充分考虑了 CPU、内存、网络带宽和磁盘 4 个维度,提出了面向能耗和效用的 VM 调度模型 VMSM-EUN,并设计了 VM 调度算法 VMSA-IPSO 来求解该模型.

3 虚拟机调度模型

本节考虑服务器上的各种硬件资源,将 VM 调度模型看作"Packing Problem". 在 VM 调度问题中,需要把服务器看作背包,VM 看作物品,在将物品放入背包时需要考虑物品的大小和背包的大小. "Packing Problem"需要一个多目标优化算法来进行求解. 本文考虑 VM 调度的 3 个目标,设计了相应的函数,从而将背包问题转换为带约束的多目标优化问题. 因此,我们需要设计目标函数,明确约束条件,建立 VM 调度模型.

3.1 相关定义

定义 1. 云中心描述. 移动云中心所有的服务器用向量 $P=\{P_1,\cdots,P_i,\cdots,P_n\}$ 表示; 移动云中心所有的 VM 用向量 $V=\{V_1,\cdots,V_j,\cdots,V_m\}$ 表示,其中 n 为移动云中心服务器数量,m 为移动云中心 VM 数量. $P_i=\{P_{cpu_i},P_{ram_i},P_{hdd_i},P_{net_i},P_{cost_i}\}$ 分别表示服务器 P_i 的 CPU 计算能力 P_{cpu_i}、内存空间 P_{ram_i}、硬盘空间 P_{hdd_i}、网络带宽 P_{net_i} 以及成本 P_{cost_i}; $V_j=\{V_{cpu_j},V_{ram_j},V_{hdd_j},V_{net_j},R_j\}$ 分别表示 VM 所需的 CPU 资源 V_{cpu_j}、内存空间 V_{ram_j}、硬盘空间 V_{hdd_j}、网络带宽 V_{net_j}; VM 效用 $R_j=\{R_{j1},\cdots,R_{ji},\cdots,R_{jn}\}$, R_{ji} 表示 VM V_j 分配到服务器 P_i 产生的效用值.

定义 2. 服务器的多维度负载度计算. 采用加权和法,将服务器中各部件的能耗占比定义为能耗权重,再通过计算加权和得到服务器的多维度负载度. 定义服务器 P_i 的多维度负载度 F_{MDL_i} 为式(1):

$$F_{MDL_i}=\omega_1\times U_{cpu_i}+\omega_2\times U_{ram_i}+\omega_3\times U_{hdd_i}+\omega_4\times U_{net_i} \quad (1)$$

$$\begin{cases} U_{cpu_i}=\dfrac{\sum_{j=1}^{m}V_{cpu_j}\times P_{ji}}{P_{cpu_i}} \\ U_{ram_i}=\dfrac{\sum_{j=1}^{m}V_{ram_j}\times P_{ji}}{P_{ram_i}} \\ U_{hdd_i}=\dfrac{\sum_{j=1}^{m}V_{hdd_j}\times P_{ji}}{P_{hdd_i}} \\ U_{net_i}=\dfrac{\sum_{j=1}^{m}V_{net_j}\times P_{ji}}{P_{net_i}} \end{cases} \quad (2)$$

其中: $U_{cpu_i},U_{ram_i},U_{hdd_i},U_{net_i}$ 分别代表服务器 P_i 的 CPU、内存、硬盘和网络带宽利用率. $\omega_1,\omega_2,\omega_3,\omega_4$ 分别为 CPU 能耗占比,内存部件能耗占比,硬盘部件能耗占比,网络部件能耗占比. $P_{ji}\in\{0,1\}$, $P_{ji}=1$ 表示 VM V_j 被部署到服务器 P_i 上运行, $P_{ji}=0$ 表示 VM V_j 未被部署到服务器 P_i 上运行.

定义 3. 虚拟机效用函数. VM 的效用函数是一个线性函数,如式(3)所示.

$$R_{vm}^{j}=\alpha_i*\ C_j+\beta_i*\ M_j+\gamma_i*\ B_j+\varphi_i*\ H_j \quad (3)$$

式(3)中的 α_i、β_i、γ_i 和 φ_i 分别表示 VM 每个单位 CPU、内存、网络带宽资源和硬盘资源的价格. C_j、M_j、B_j 和 H_j 表示 VM V_j 的 CPU、内存、网络带宽和硬盘的资源配置大小.

定义 4. 服务器效用函数. 一台服务器的效用值为该服务器上所有 VM 产生的效用与该服务器的成本的差值,因此服务器的效用函数如式(4)所示.

$$R_{server}^{i}=\sum_{j=1}^{m}R_{vm}^{ji}-P_{cost}^{i} \quad (4)$$

式(4)中的 m 表示该服务器上的 VM 数量, R_{vm}^{ji} 表示服务器 P_i 上 VM V_j 的效用, P_{cost}^{i} 表示服务器 P_i 的成本.

云中心的效用为所有物理机的效用值之和,因此云中心的效用函数如式(5)所示.

$$R_{total}=\sum_{i=1}^{n}R_{server}^{i} \quad (5)$$

3.2 约束条件

为找到 VM 和服务器之间的最佳映射方案,实现云中心的效益最大化和低能耗的目标,在进行 VM 调度时需要明确约束条件. 下面给出 VM 调度模型需满足的约束条件:

$$s.t$$

$$\begin{cases} \sum_{i=0}^{n}P_{ji}\leqslant 1,\forall j\in\{1,2,\cdots,m\} \\ \sum_{j=0}^{m}V_{cpu_j}\times P_{ji}\leqslant P_{cpu_i},1\leqslant i\leqslant n \\ \sum_{j=0}^{m}V_{ram_j}\times P_{ji}\leqslant P_{ram},1\leqslant i\leqslant n \\ \sum_{j=0}^{m}V_{hdd_j}\times P_{ji}\leqslant P_{hdd},1\leqslant i\leqslant n \\ \sum_{j=0}^{m}V_{net_j}\times P_{ji}\leqslant P_{net_i},1\leqslant i\leqslant n \end{cases} \quad (6)$$

式(6)中第一个式子表示对于移动云数据中心中的任意一台 VM,在任意时刻它只能运行在某一台服务器上运行; 式(6)中其余的不等式表示一台服务器所承载 VM 占用的资源总和小于等于其所拥有的资源.

3.3 目标函数

为实现移动云数据中心效益最大化和降低运行成本的目标,本小节根据上述定义与调度约束条件,设计了 3 个 VM 调度的目标函数,具体如下.

3.3.1 最小能耗

文献[19]提出了一个新的服务器的能耗模型,并证明了服务器的能量消耗与 CPU 利用率呈线性关系. 本文在该文献提出的服务器能耗模型的基础上进行优化,不是仅仅考虑 CPU 利用率而是将服务器的能耗与服务器的负载相联系,考虑了 CPU、内存、网络带宽和磁盘 4 个维度,提出了改进的服务器能耗模型. 首先将服务器 P_i 的多维度负载度 F_{MDL_i} 进行归一化处理,得到服务器的多维度负载率 U_{MDL_i},归一化处理方法如式(7)所示. 其中 $F\min_{MDL_i}$, $F\max_{MDL_i}$ 表示优化周期 $[t_1,t_2]$ 内服务器 P_i 的多维负载度的最小值和最大值.

$$U_{MDL_i}=\frac{F_{MDL_i}-F\min_{MDL_i}}{F\max_{MDL_i}-F\min_{MDL_i}} \quad (7)$$

改进的云服务器能耗模型,如式(8)-式(10)所示.

$$P(U_{MDL_i}(t)) = c \times P_{\max_i} + (1-c) \times P_{\max_i} \times U_{MDL_i}(t) \quad (8)$$

$$E_i = \int_{t_1}^{t_2} P(U_{MDL_i}(t))\ dt \quad (9)$$

$$E = \sum_{i=1}^{n} E_i \times Y_i, \text{with} \quad Y_i = \begin{cases} 1 & \sum_{j=1}^{m} P_{ji} \geqslant 1 \\ 0 & \sum_{j=1}^{m} P_{ji} = 0 \end{cases} \quad (10)$$

其中,式(8)将服务器的能耗与服务器的负载相联系,考虑了CPU、内存、网络带宽和磁盘4个维度,而不是仅仅考虑CPU利用率. $P_{\max_i}$表示服务器P_i的多维负载率达到最高峰值时的最大功耗; $U_{MDL_i}(t)$表示服务器P_i在t时刻的多维负载率. 式(9)中E_i表示优化周期$[t_1,t_2]$内服务器P_i的能耗. 式(10)中,E表示移动云中心所有服务器的能耗,单位为瓦特(W),n表示移动云数据中心的服务器数量; $Y_i \in \{0,1\}$表示服务器P_i是否处于激活状态. $Y_i=1$,表示服务器P_i处于激活状态. $Y_i=0$,表示服务器P_i处于关闭状态.

3.3.2　最大效用

移动云计算中心的效用为全部VM效用和所有服务器成本的差,计算方法如式(11)所示.

$$R = \sum_{j=1}^{m}\left(\sum_{i=1}^{n} R_{ji} \times P_{ji}\right) - \sum_{i=1}^{n} P_{Cost_i} \times Y_i, \text{with } Y_i = \begin{cases} 1 & \sum_{j=1}^{m} P_{ji} \geqslant 1 \\ 0 & \sum_{j=1}^{m} P_{ji} = 0 \end{cases} \quad (11)$$

式(11)中R表示移动云中心的总效用值; n表示移动云中心服务器总数; m表示云中心承载的VM总数; R_{ji}表示在服务器P_i上运行VM V_j产生的VM效用; P_{cost_i}表示服务器P_i的使用成本.

3.3.3　最小服务器数

通常在虚拟资源调度的优化阶段,为了进一步降低能耗,需要将资源利用率较低的服务器上的VM迁移到其它服务器上,并将该服务器设为休眠状态,使得云中心可以运行较少的服务器来承载相同的负载,以进一步减少数据中心的能量消耗. 服务器数计算方法如式(12)所示.

$$NUM_{servers} = \sum_{i=1}^{n} Y_i, \text{with} \quad Y_i = \begin{cases} 1 & \sum_{j=1}^{m} P_{ji} \geqslant 1 \\ 0 & \sum_{j=1}^{m} P_{ji} = 0 \end{cases} \quad (12)$$

综上所述,VM调度模型VMSM-EUN可以描述为:

$$\min: E = \sum_{i=1}^{n} E_i \times Y_i \quad (13)$$

$$\max: R = \sum_{j=1}^{m}\left(\sum_{i=1}^{n} R_{ji} \times P_{ji}\right) - \sum_{i=1}^{n} P_{Cost_i} \times Y_i \quad (14)$$

$$\min: NUM_{servers} = \sum_{i=1}^{n} Y_i \quad (15)$$

$$s.t$$

$$\begin{cases} \sum_{i=0}^{n} P_{ji} \leqslant 1, \forall j \in \{1,2,\cdots,m\} \\ \sum_{j=0}^{m} V_{cpu_j} \times P_{ji} \leqslant P_{cpu_i}, 1 \leqslant i \leqslant n \\ \sum_{j=0}^{m} V_{ram_j} \times P_{ji} \leqslant P_{ram}, 1 \leqslant i \leqslant n \\ \sum_{j=0}^{m} V_{hdd_j} \times P_{ji} \leqslant P_{hdd}, 1 \leqslant i \leqslant n \\ \sum_{j=0}^{m} V_{net_j} \times P_{ji} \leqslant P_{net_i}, 1 \leqslant i \leqslant n \end{cases} \quad (16)$$

4　基于改进粒子群的虚拟机调度算法

4.1　改进的粒子群优化算法

粒子群优化(Particle Swarm Optimization,PSO)算法是模拟鸟群寻找食物的集体行为而提出的一种基于群体智能的全局优化技术. PSO中成员粒子以一定的速度和加速度向更好的位置移动,每个粒子表示问题空间的解[18].

在PSO中,每一个粒子被认为是潜在的解决方案,每个粒子都与两个向量相关联,它们分别是速度向量$v(t)=[v^1,v^2,\cdots,v^D]$和位置向量$x(t)=[x^1,x^2,\cdots,x^D]$,其中$D$为解空间的维度. 在进化过程中改进的PSO算法的更新方程如式(17)、式(18)[20]所示,在PSO中所有粒子是根据当前局部最优解$pbest$和当前全局最优解$gbest$来更新自己的信息.

$$v(t+1) = \omega v(t) + c_1 r_1 (pbest - x(t)) + c_2 r_2 (gbest - x(t)) \quad (17)$$

$$x(t+1) = x(t) + v(t+1) \quad (18)$$

其中,v为粒子速度,ω为惯性因子,c_1、c_2为加速因子,r_1、r_2为(0,1)上一致分布的随机数. 每一维上的粒子速度被限制在$v_{\max}(v_{\max}>0)$内,如果某一维更新后的速度大于用户给定的$v_{\max}$,那么就设为$v_{\max}$,即$v(t+1)>v_{\max}$时,$v(t+1)=v_{\max}$. 如果某一维更新后的速度小于等于用户给定的$-v_{\max}$,那么就设为$-v_{\max}$,即$v(t+1)\leqslant -v_{\max}$时,$v(t+1)=-v_{\max}$. 目前已有的研究对PSO进行了改进,典型工作有主要3个方面: YH Shi等人将惯性因子引入PSO,并证明了当惯性因子值较大时,全局寻优能力强,局部寻优能力弱; 值较小反之[21]. 除惯性因子外,加速因子c_1和c_2也是PSO中的重要参数. 权重因子包括惯性因子ω和加速因子c_1、c_2,使粒子保持着原有的运动状态,并具有探索和开发的能力. 在Kennedy的两个极端情况下,实验表明"仅社交"模型和"仅认知"模型中,两种加速因子对于PSO的成功至关重要[22]. Kennedy和Eberhart建议c_1、c_2固定值为2.0,这种配置已被许多研究人员采用[23]. Suganthan证明了对于不同的问题,使用c_1、c_2的"ad hoc"值而不是固定值可以产生更好的性能[24]. Ratnaweera等人提出了一种具有线性时变加速度系数(HPSO-TVAC)的PSO算法,在开始时设置较大的c_1和较小的c_2,并且在搜索期间逐渐反转. 在对比实验中,HPSO-TVAC显示出最佳的整体性能. 这是由于时变c_1和c_2可以平衡全局搜索能力和局部搜索能力[25]. 由上所述可见,在粒子群寻优的过程中采用适合的惯性因子ω、加速因子c_1和c_2,将有助于提高PSO性能. 因此我们在4.3小节中,将在粒子群求解过程中确定每一次迭代后粒子群所处的状态,动态自适应的改变惯性因子ω和加速因子c_1、c_2的取值.

4.2　映射关系与编码方式

为了将PSO算法用于求解VM调度问题,我们需要将粒子的编码与现实问题对应,本文中粒子的位置、速度与现实中VM调度问题的映射关系如下所述.

4.2.1　粒子i在t时刻的位置X_i^t

粒子i在t时刻的位置被定义为n维向量$X_i^t=(x_{i1}^t,x_{i2}^t,\cdots,x_{in}^t)$,表示一个VM部署方案,其中$n$为云数据中心服务器数量. 向量$X_i^t$中每一维的值为0或1,每一维都对应一个物理机到VM的映射方案(一对多的映射关系),当值为1时,

表明该服务器是处于激活状态的,这时在该服务器上会部署一些 VM,形成一对多的映射. 当值为 0 时,则不部署 VM,编码方式如图 1 所示.

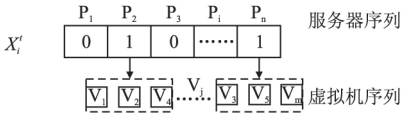

图 1 编码方式

Fig. 1 Encoding scheme

4.2.2 **粒子 i 在 t 时刻的速度 V_i^t**

粒子 i 在 t 时刻的速度被定义为 n 维向量 $V_i^t=(v_{i1}^t,v_{i2}^t,\cdots,v_{in}^t)$,表示 VM 部署位置是否需要调整. 向量 V_i^t 中每一位的值为 0 或 1. 如果必须重新评估和调整相应的服务器及其 VM,则该值为 0,否则值为 1. V_i^t 能使 VM 部署调整到最佳解决方案,引导粒子位置更新操作.

4.3 自适应参数调整

考虑 PSO 的特性,我们将粒子群求解过程中状态变化与加速因子 c_1 和 c_2 的取值联系在一起,自适应的调整 c_1 和 c_2. 并且将粒子群的进化因子 f 与惯性因子 ω 建立映射关系,使得 ω 的取值跟随进化状态的变化而变化. 具体的自适应参数调整方法如下所述.

4.3.1 **加速因子 c_1 和 c_2 的调整**

设 d_i 为第 i 个到其余 $i-1$ 个粒子的平均距离,即 $d_i=\frac{1}{N-1}\sum_{j=1,j\neq i}^{N}\sqrt{\sum_{k=1}^{D}(x_i^k-x_j^k)^2}$. 其中 N 为粒子群大小,D 为维数; 定义全局最优粒子的 d_i 为 d_g. 设 $f=\frac{d_g-d_{\min}}{d_{\max}-d_{\min}}\in[0,1]$,其中 $d_{\max}$ 为所有平均距离中的最大值,$d_{\min}$ 为所有平均距离中的最小值.

参考文献[26]: 设 f 为进化因子. 将粒子群的状态分类为: 探索、开发、收敛和跳出 4 种状态. 在粒子群求解过程中通过计算进化因子 f 的数值大小来确定每一次迭代后粒子群所处的状态,最后得出进化因子 f 与粒子群所处的状态的模糊隶属函数. 模糊隶属函数有交叉的区域,即存在一个 f 对应两个函数值. 我们称这个区域为过渡期. 在过渡期,粒子群可能处于两种状态中的任意一种. 对于最终分类,可以使用“单例法”或“质心法”[27] 两种常用的去模糊技术中的任一种. 本文使用单例法,因为它比质心法更有效,并且与去模糊规则表一起实现起来比较简单. 与大多数模糊逻辑方案类似,我们设计的去模糊规则表也涉及状态和“状态变化”变量. 探索、开发、收敛和跳出 4 种状态分别用 S_1、S_2、S_3 和 S_4 表示. 根据参考文献[26],PSO 序列 $S_1\Rightarrow S_2\Rightarrow S_3\Rightarrow S_4\Rightarrow S_1\Rightarrow\cdots$ 反映了 PSO 过程中状态的变化. 模糊隶属函数图像的交叉区域分别为: S_1 与 S_2 交叉区域、S_2 与 S_3 交叉区域、S_1 与 S_4 交叉区域. 为了分类的稳定性即不过度的切换状态指示符,设计了如表 1 所示的去模糊化规则表. 表 1 中,$F(f)$ 表示模糊隶属度,S_{t-1} 表示前一个状态,S_t 表示当前状态的选取结果.

在粒子群优化的状态模糊分类后,我们将根据每一次迭代后粒子群所处的状态,动态的改变加速因子 c_1 和 c_2 的取值. 综上所述,我们得到以下调整系数 c_1 和 c_2 的策略:

1) 在探索状态下增大 c_1 和减小 c_2: 在探索状态下探索尽可能多的最佳值非常重要,增大 c_1 和减小 c_2 可以帮助粒子单独探索并达到它们自己的历史最佳位置,而不是围绕当前可能与局部最优相关的最佳粒子.

表 1 去模糊规则表

Table 1 Fuzzy control rules

S_{t-1} / $F(f)$ / S_t	S_1	S_2	S_3	S_4
存在 S_1 与 S_2 两个隶属度	S_1	S_2	S_2	S_1
存在 S_2 与 S_3 两个隶属度	S_2	S_2	S_3	S_2
存在 S_1 与 S_4 两个隶属度	S_1	S_1	S_4	S_4

2) 在开发状态下略微增大 c_1 并略微减小 c_2: 在这种状态下,粒子利用局部信息并聚集到每个粒子的历史最佳位置所指示的可能的局部最佳位置. 因此,缓慢增加 c_1 并保持相对较大的值可以增强围绕局部最好 $pbest$ 的搜索和利用. 与此同时,全局最佳粒子并不总是在此阶段定位全局最优区域. 因此,缓慢减少 c_2 并保持较小值可以避免陷入局部最优. 此外,在探索状态之后和收敛状态之前更可能发生开发状态. 因此,改变 c_1 和 c_2 的时间应该是从探索状态略微改变到收敛状态.

3) 在收敛状态下略微增大 c_1 并略微增大 c_2: 在收敛状态下,群体似乎找到全局最优区域,因此,应增强 c_2 的影响以将其他粒子引导到可能的全局最优区域. 因此,应增大 c_2 的值. 另一方面,应减小 c_1 的值以使群快速收敛,但是这种策略会过早地将这两个参数饱和到它们的下限和上限. 结果是群体将被当前最佳区域强烈吸引,导致过早收敛,如果当前最佳区域是局部最优,则是有害的. 为避免这种情况,c_1 和 c_2 都略有增加. 略微增大两个加速因子最终将具有与减小 c_1 和增大 c_2 相同的预期效果,因为它们的值将被限制到大约 2.0,因为 c_1 和 c_2 之和的上限为 4.0[23]. 当 c_1 和 c_2 的和大于 4.0 时,需要对 c_1 和 c_2 进行归一化,归一化处理方法如式(19)所示.

$$c_i=\frac{c_i}{c_1+c_2}\times 4.0,\quad i=1,2. \tag{19}$$

4) 在跳出状态下减小 c_1 和增大 c_2: 当全局最佳粒子跳出局部最佳位置并朝向更好的最佳状态时,它可能远离拥挤群集. 一旦这个新区域被一个粒子找到,这颗粒子就会成为(可能是新的)领导者,其他粒子应该跟随它并尽可能快地趋近这个新区域. 较大的 c_2 和相对较小的 c_1 有助于实现这一目的.

4.3.2 **惯性因子 ω 的调整**

PSO 中的惯性因子 ω 用于平衡全局和本地搜索功能. 许多研究人员主张 ω 的值在探索状态下应该很大而在开发状态下应该很小. 然而,纯粹随着时间减少 ω 并不一定正确. 进化因子 f 与惯性权重 ω 具有一些特征,其中 f 在探索状态期间也相对较大并且在收敛状态下变得相对较小. 因此,ω 的取值应跟随进化状态的变化而变化,即令 ω 与 f 存在这样的映射 $\omega(f):R^+\rightarrow R^+$.

$$\omega(f)=\frac{2}{2+3e^{-2.6f}}\in[0.4,0.9]\quad\forall f\in[0,1] \tag{20}$$

在本文中,ω 初始化为 0.9. 由于 ω 不一定随时间单调变化,而是随 f 单调变化,因此 ω 将适应以 f 为特征的搜索环境. 在跳出或探索状态下,较大 f 和 ω 将有利于全局搜索. 相反,当 f 很小时,检测到开发或收敛状态,ω 减小以有利于局

部搜索. 设计的自适应参数调整流程图如图 2 所示.

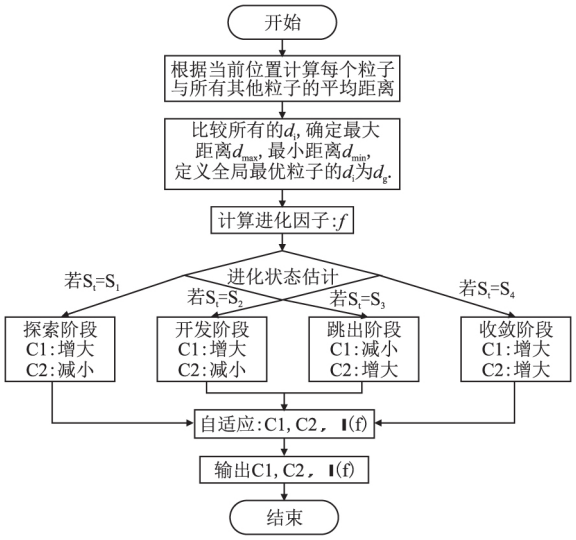

图 2　自适应参数调整流程图

Fig. 2　Self-adaptive parameter adjustment flow chart

4.4　虚拟机调度算法

基于改进粒子群,我们设计了 VM 调度算法 VMSA-IPSO,算法具体步骤如下:

Step 1. 定期收集连续到达的 VM 请求作为算法的输入;

Step 2. 初始化.

Step 2-1. 根据 VMSM-EUN 模型中决策变量的个数,设置种群大小为 N,最大迭代次数为 Gen.

Step 2-2. 生成初始粒子群. 根据 VM 请求以及服务器的各种资源约束,随机生成 N 个可行 VM 部署方案,每一个解都是一个粒子,每个粒子都是由二维编码方案编码的,这些粒子构成了初始种群. 粒子的初始位置由初始种群决定,粒子的初始速度由粒子的第一维状态信息决定.

Step 3. 计算每个粒子到其他粒子的平均距离和进化因子 f 根据进化因子 f 的大小对粒子群的状态进行评估,自适应的调整参数 c_1、c_2 和 ω .

Step 4. 粒子群中为一个粒子,根据式(10)、式(11)和式(12)计算其目标函数的值.

Step 5. 对于每个粒子更新其当前最好位置 *pbest*.

Step 6. 比较所有粒子的当前最好位置 *pbest*,选出适应值最高的与当前全局最好位置 *gbest* 作比较,得出种群的全局最佳位置 *gbest*.

Step 7. 根据式(17)更新种群中所有粒子的速度.

Step 8. 更新种群中所有粒子的位置.

Step 8-1. 根据式(18)更新粒子位置,此位置更新操作可能会丢失 VM,将在后续步骤中重新部署丢失的 VM.

Step 8-2. 删除 VM. 上述位置更新操作可能导致 VM 部署在两台服务器上. 因此,必须删除重复的 VM 以确保部署方案的可行性. 删除方法是将二维编码的本地位置设置为 0,并将这些重复的 VM 一起删除. 同样,这些被删除的 VM 将在下一步重新部署.

Step 8-3. 重新部署被删除的 VM. 要获得更新后的可行解决方案,就必须将丢失或意外删除的 VM 重新部署到服务器上. 首先将当前处于激活状态的服务器作为主机服务器,当所有处于激活状态的服务器都无法放置 VM 时才会启用新服务器.

Step 9. 根据更新后的粒子群,更新局部最优粒子和全局最优粒子的位置信息.

Step 10. 判断是否达到最大迭代次数,若达到则输出 VM 调度方案,否则重复步骤 Step 3 ~ Step 9.

算法 1. VM scheduling algorithm based on improved particle swarm optimization

输入: Virtuel machine request sequence

输出: The optimal particle, Virtual machine deployment plan.

1. Initialize the population size N, the maxinmum number of iterations Gen. Set $c_1 = 2.0$. $c_2 = 2.0$. $\omega = 0.9$
2. **for** *each particle* **do**//Generate an initial particle swarm, N feasible virtual machine deployments.
3. Initialize particle//Initialize velocity and position of particle.
4. **end for**
5. Set $gen = 0$.
6. **while** $gen < Gen$ **do**
7. Calculate the evolution factor f.
8. Estimating the evolutionary state of particle swarm.
9. Adjusttng parameters adaptively. //Adjust c_1, c_2, ω.
10. Set $i = 0$.
11. **while** $i < N$ **do**
12. Calculate the value of the objective functions for particle i based on(10), (11), and(12).
13. Update the particle i's *pBest*
14. $i = i + 1$.
15. **end while**
16. Compare all paricles, get the global best position *gBest*.
17. Updating the velocity of all particles according to(17).
18. **for** *each particle* **do**
19. Updating the position of particle according to(18).
20. Removing virtual machines.
21. Redeploying the virtual machines.
22. **end for**
23. Updating the location information of local and global optimal particles according to the updated particle swarm.
24. $gen = gen + 1$.
25. **end while**

图 3　基于改进粒子群的虚拟机调度算法伪代码

Fig. 3　Pseudo code of the VM scheduling algorithm

基于改进粒子群的 VM 调度算法 VMSA-IPSO 的伪代码,如图 3 所示.

5　实验与结果分析

本文基于多目标优化算法建立了 VM 调度模型 VMSM-EUN,将最小化数据中心能耗、最大化数据中心效用以及最小化服务器数作为目标,设计并实现了基于改进粒子群的 VM 调度算法 VMSA-IPSO 来求解该模型.

5.1　实验参数设置

为了验证 VMSM-EUN 模型和 VMSA-IPSO 算法的有效性,本文使用云计算仿真工具 CloudSim 进行仿真实验. 为验

证本文提出的 VM 资源调度算法的性能,在相同环境和条件下将本文提出的 VM 调度算法 VMSA-IPSO 与文献[29]提出的 MBFD(Modified Best Fit Decreasing algorithm)算法以及"Packing Problem"问题近似算法 FF(First-Fit algorithm)、BF(Best-Fit algorithm)进行比较. 从活动服务器数量、服务器效用和能耗 3 个方面进行了对比分析. 实验模拟包含 400 个异构服务器的异构虚拟化数据中心,为了反映虚拟化数据中心的异构性,我们根据参考文献[30],选取了两种类型的服务器,它们具有不同的配置和能耗特征,服务器的参数特征如表 2 所示. 所选服务器在不同负载级别的功耗(瓦特)如表 3[31]所示. 对比仿真实验的参数设置如表 4 所示. 根据文献[32],在基于 Xeon 处理器的服务器中,CPU 是能耗最大的部件,其能耗占比高达 74%,内存能耗占比为 19%,硬盘能耗占比为 5%,网络部件能耗占比 2%. 在表 4 中 ω_1,ω_2,ω_3,ω_4 分别为 CPU 能耗权重,内存能耗权重,硬盘能耗权重,网络带宽能耗权重. 表 5 给出了本次实验的 VM 实例.

表 2 服务器参数特征

Table 2 Server configuration

服务器类型	CPU	内存
HP ProLiant G4	Intel Xeon 3040 2cores* 1860MHz	4G
HP ProLiant G5	Intel Xeon 3075 2cores* 2660MHz	8G

表 3 服务器在不同负载级别的功耗(瓦特)

Table 3 Power consumption of servers at various load levels(W)

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

表 4 实验参数配置

Table 4 Experimental parameter configuration

粒子群规模	迭代次数	ω_1	ω_2	ω_3	ω_4
30	60	0.74	0.19	0.05	0.02

表 5 虚拟机实例

Table 5 Virtual machine instances

虚拟机类型	CPU(MIPS)	内存(MB)	网络带宽(Mbps)	α_i	β_i	γ_i	φ_i
Instance1	500	613	100	4	4	4	2
Instance2	1000	1700	100	8	8	8	2

5.2 活动服务器数量比较

在本文中,活动服务器的数量是处于活动状态的服务器总数,这些服务器运行承载云服务的 VM. 在此实验中,VM 请求的数量为 100 到 1000,我们将激活 HP ProLiant G4 服务器或 HP ProLiant G5 服务器来响应其请求. 图 4 给出了与其他方法的对比实验结果.

由图 4 可见,随着 VM 请求数量的增加,本文提出的算法总是能激活最小数量的服务器,而 FF 算法始终激活最大数量的服务器. MBFD 算法激活的服务器总数小于 BF 算法和 FF 算法,但激活的服务器总数多于本文提出的算法 VMSA-IPSO. 因此,VMSA-IPSO 算法能够使移动云数据中心开启更少的服务器,提高了资源的利用率.

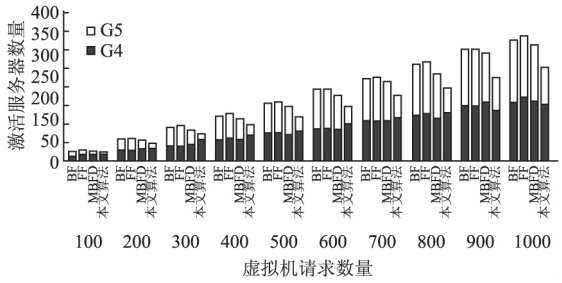

图 4 激活服务器总数

Fig. 4 Total number of active servers

5.3 数据中心效用比较

在本文中,数据中心的全局效用定义为所有 VM 的效用与所有服务器运营成本的差值. 根据式(4)、式(5)可得:

$$R_{total}=\sum_{i=1}^{n}R_{server}^{i}=\sum_{i=1}^{n}\left(\sum_{j=1}^{m}R_{vm}^{ji}-P_{cost}^{i}\times Y_i\right)$$
$$=\left(R_{vm}^{11}+R_{vm}^{12}+\cdots+R_{vm}^{21}+R_{vm}^{22}+\cdots\right)-\left(C_{cost}^{1}\times Y_1+C_{cost}^{2}\times Y_2+\cdots\right)=R_{vm}-P_{cost} \quad (21)$$

式(21)中,R_{total}、R_{vm}和 P_{cost}分别为数据中心的全局效用、所有 VM 的效用和所有服务器的成本. $Y_i\in\{0,1\}$表示服务器 P_i 是否处于激活状态. $Y_i=1$,表示服务器 P_i 处于激活状态. $Y_i=0$,表示服务器 P_i 处于关闭状态. 根据式(21),我们可以计算在不同 VM 请求下,各个 VM 调度算法产生的全局效用值,计算结果如图 5 所示.

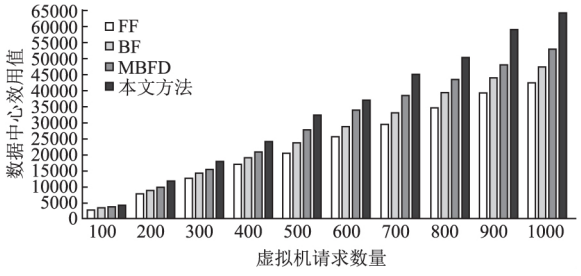

图 5 数据中心效用值比较

Fig. 5 Data center utility values comparison

如图 5 所示,与其他算法相比,我们的算法获得了更高的效用值. 随着 VM 请求数量的增长,云中心在这 4 种算法下的效用值各不相同,但是本文提出方法的效用值总是高于 FF、BF 和 MBFD. 对于同一组 VM 请求,如果虚拟化数据中心中服务器的利用率较高,可激活较少的服务器来承载云服务工作负载,此时虚拟化数据中心的能源消耗将减少,数据中心的效用会大幅度增加.

5.4 能耗比较

在本文中,能耗是指所有处于激活状态的服务器的总能耗,计算方法如式(8)-式(10)所示. 在满足不同数量的 VM

请求下,对比的各个算法的总能耗如图 6 所示.

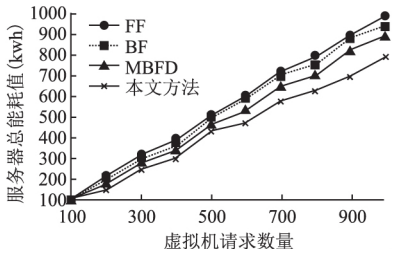

图 6　总能耗对比图

Fig. 6　Total energy consumption comparison chart

通过图 6 我们可以得出,无论 VM 请求的规模如何,本文提出的方法均可使数据中心运营商能够节省更多能量. 与其他 3 种方法相比,我们的方法可以节省较多能源费用. 这是因为 FF,BF 和 MBFD 在解决问题的过程中,缺乏虚拟化数据中心中异构服务器的能耗特征等反映全局状态的信息. 那些算法只考虑了多维资源约束,且未考虑不同服务器的能量差异.

5.5　VMSA-IPSO 算法性能分析

为比较 VMSA-IPSO 算法的收敛时间,本文将未引入参数自适应调整策略的调度算法与 VMSA-IPSO 算法进行比较. 在对比实验中,粒子群大小为30,迭代次数为100. 图7中,

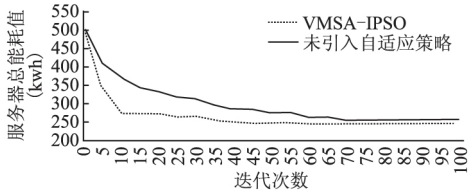

图 7　收敛时间对比 1(虚拟机请求数量: 300)

Fig. 7　Convergence time comparison 1

(Number of virtual machine requests: 300)

VM 请求数量为 300. 图 8 中,VM 请求数量为 800. 从图 7 和图 8 可以看出,VMSA-IPSO 开始进化并快速收敛,然后趋向平稳,能达到很好的收敛效果,收敛时间大约为 50 ~ 60 次迭代的时间. 而未引入参数自适应调整策略的算法的收敛时间大约为 75-85 次迭代的时间.

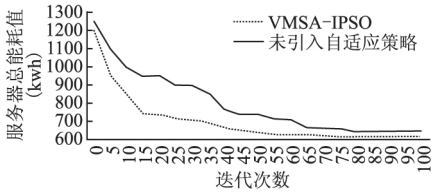

图 8　收敛时间对比 2(虚拟机请求数量: 800)

Fig. 8　Convergence time comparison 2

(Number of virtual machine requests: 800)

综上所述,本文方法引入关键参数自适应调整策略,增强了 VMSA-IPSO 调度算法的收敛性,加快了算法的求解速度,使其能够更快找到更好的 VM 调度方案,提高调度方案的质量.

6　结　论

本文以最小化能耗、最大化效用以及最小化服务器数为目标,建立了 VM 调度模型 VMSM-EUN,设计了基于改进粒子群的 VM 调度算法 VMSA-IPSO 来求解该模型. 本文在综合考虑了云中心效用和能耗的前提下,提出的调度算法能更好的解决移动云计算环境下的 VM 调度问题. 通过与现有算法的对比实验,验证了本文提出的基于改进粒子群的 VM 调度算法具有使云中心能耗更低,效用更大的优点,能进行更优的 VM 调度.

References:

[1] Yeganeh H,Salahi A,Pourmina M A. A novel cost optimization method for mobile cloud computing by capacity planning of green data center with dynamic pricing [J]. Canadian Journal of Electrical and Computer Engineering-Revue Canadienne De Genie Electrique Et Informatique,2019,42(1):41-51.

[2] Zhou B,Srirama S N,Buyya R. An auction-based incentive mechanism for heterogeneous mobile clouds [J]. Journal of Systems and Software,2019,152:151-164,doi:10.1016/j.jss.2019.03.003.

[3] Dinh H T,Lee C,Niyato D,et al. A survey of mobile cloud computing: architecture,applications,and approaches [J]. Wireless Communications & Mobile Computing,2013,13(18):1587-1611.

[4] Abolfazli S,Sanaei Z,Gani A,et al. Rich mobile applications: genesis,taxonomy,and open issues [J]. Journal of Network & Computer Applications,2014,40(7):345-362.

[5] Khan A U R,Othman M,Xia F,et al. Context-aware mobile cloud computing and its challenges [J]. IEEE Cloud Computing,2015,2(3):42-49.

[6] Sanaei Z,Abolfazli S,Gani A,et al. Heterogeneity in mobile cloud computing: taxonomy and open challenges [J]. IEEE Communications Surveys & Tutorials,2014,16(1):369-392.

[7] Li X,Garraghan P,Jiang X,et al. Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy [J]. IEEE Transactions on Parallel & Distributed Systems,2018,29(6):26-31.

[8] Su Yu,Gao Yang,Qin Zhi-guang. Power-aware VM dynamic mapper using particle swarm optimization [J]. Computer Science,2015,42(12):26-31.

[9] Lago D G,Madeira E R M,Medhi D. Energy-aware virtual machine scheduling on data centers with heterogeneous bandwidths [J]. IEEE Transactions on Parallel & Distributed Systems,2018,29(1):83-98.

[10] Li Y,Chen M,Dai W,et al. Energy optimization with dynamic task scheduling mobile cloud computing [J]. IEEE Systems Journal,2017,11(1):96-105.

[11] Khan A A,Zakarya M,Khan R. H2-a hybrid heterogeneity aware resource orchestrator for cloud platforms [J]. IEEE Systems Journal,2019,13(4):3873-3876.

[12] Jang J,Jung J,Hong J. An efficient virtual CPU scheduling in cloud computing [J]. Soft Computing,2020,24(8):1-11.

[13] Yang T,Pen H,Li W,et al. An energy-efficient virtual machine placement and route scheduling scheme in data center networks [J]. Future Generation Computer Systems,2017,77:1-11,https://doi.org/10.1016/j.future.2017.05.047.

[14] Guo W X,Kuang P,Jiang Y Q,et al. SAVE: self-adaptive consolidation of virtual machines for energy efficiency of CPU-intensive applications in the cloud [J]. Journal of Supercomputing,2019,75(11):7076-7100.

[15] Xu H, Yang L, Wei W, et al. Incentive-aware virtual machine scheduling in cloud computing [J]. Journal of Supercomputing, 2018, 74(4): 1-23.

[16] Zaman S, Grosu D. A combinatorial auction-based mechanism for dynamic VM provisioning and allocation in clouds [J]. IEEE Transactions on Cloud Computing, 2014, 1(2): 129-141.

[17] Liu X, Yuan C-W, Yang Z, et al. A scheme of virtual machine pricing and allocation in mobile cloud computing [J]. Journal of University of Electronic Science and Technology of China, 2016, 45(2): 197-201.

[18] Zhong Wei. Research on virtual resource management technology in mobile cloud computing [D]. Nanjing: Nanjing University of Aeronautics and Astronautics, 2018.

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

[20] Xue F Y, Lv X L, Chai H M, et al. Application of particle swarm optimization to the estimation of the TSInSAR deformation parameter [J]. Remote Sensing Letters, 2019, 10(8): 756-765.

[21] Shi Y H, Eberhart R C. A modified particle swarm optimizer [C]// IEEE International Conference on Evolutionary Computation, 1998: 69-73.

[22] Kennedy J. The particle swarm: social adaptation of knowledge [C]//IEEE International Conference on Evolutionary Computation, 1997: 303-308.

[23] Kennedy J, Eberhart R. Particle swarm optimization [C]//Proc. of 1995 IEEE Int. Conf. Neural Networks, (Perth, Australia), 2011, 4(8): 1942-1948.

[24] Suganthan P N. Particle swarm optimiser with neighbourhood operator [C]//Proceedings of the Congress on Evolutionary Computation, 1999: 1958-1962.

[25] Ratnaweera A H S K, Watson H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients [M]. IEEE Press, 2004.

[26] Zhan Z H, Zhang J, Li Y, et al. Adaptive particle swarm optimization [J]. IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics, 2009, 39(6): 1362-1381.

[27] Jang J S R, Sun C T, Mizutani E. Neuro-fuzzy and soft computing; a computational approach to learning and machine intelligence [J]. IEEE Transactions on Automatic Control, 1997, 42(10): 1482-1484.

[28] Zhan Z H, Jing X, Zhang J, et al. Adaptive control of acceleration coefficients for particle swarm optimization based on clustering analysis [C]//IEEE Congress on Evolutionary Computation, 2007: 3276-3282.

[29] Xin L, Zhuanzhuan Z. A virtual machine dynamic migration scheduling model based on MBFD algorithm [J]. International Journal of Computer Theory and Engineering, 2015, 7(4): 278-282.

[30] Shaw R, Howley E, Barrett E. An energy efficient anti-correlated virtual machine placement algorithm using resource usage predictions [J]. Simulation Modelling Practice and Theory, 2019, 93: 322-342, doi: 10.1016/j.simpat.2018.09.019.

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

[32] Dayarathna M, Wen Y, Fan R. Data center energy consumption modeling: a survey [J]. IEEE Communications Surveys & Tutorials, 2016, 18(1): 732-794.

附中文参考文献:

[8] 苏 宇,高 阳,秦志光. 功耗感知的自适应粒子群优化虚拟机动态映射 [J]. 计算机科学,2015,42(12):26-31.

[18] 钟 伟. 移动云计算环境下虚拟资源管理技术的研究 [D]. 南京: 南京航空航天大学,2018.