DOI:10.13718/j.cnki.xsxb.2021.03.002

基于改进萤火虫算法的云计算任务调度策略①

朱利华[1], 朱玲玲[2]

1. 常州信息职业技术学院 软件与大数据学院,江苏 常州 213164;
2. 南通大学 信息科学技术学院,江苏 南通 226200

摘要:针对云计算中任务调度效率低引起的资源利用不平衡问题,提出一种基于改进萤火虫算法的虚拟机任务调度策略,该策略首先构建云计算资源负载平衡优化问题的约束条件,采取最小用户任务完成时间作为资源优化的目标函数;其次通过改进的萤火虫算法优化资源搜索路径,优化云服务器中多个虚拟机之间的任务负载平衡,通过提高云服务器的响应效率达到缩短用户任务完成总时间的目的. 实验结果表明:相比于其他算法,本文提出的策略在云计算任务完成时间方面具有明显优势,能够有效地解决服务器中的负载不平衡问题,提高用户请求的响应效率.

关 键 词:云计算;任务调度;萤火虫算法;负载均衡

中图分类号:TP391 **文献标志码**:A **文章编号**:1000-5471(2021)03-0007-06

云计算是目前计算机领域内技术发展的焦点之一,它是将传统的分布式计算、并行计算及网格计算融为一体的数据服务系统[1]. 美国国家标准与技术研究院定义:云计算是一种按使用量付费的模式,这种模式为用户提供低成本、可伸缩、高可靠性的计算资源及服务[2-3]. 目前,按照云计算可提供的服务种类,划分为软件即服务(Software as a service, SaaS)、平台即服务(Platform as a Service, PaaS)、基础设施即服务(Infrastructure as a Service, IaaS)[4]. 在云环境中,计算资源服务质量的好坏是衡量云计算效果的一个重要指标,因此如何有效地降低云计算中的负载均衡,提升计算资源利用率是当前研究的热点问题.

负载均衡是指将任务分摊到多个操作单元上执行的过程,避免产生系统瓶颈或者资源浪费[5]. 迄今为止,国内外研究学者已经开发出许多传统算法、博弈论算法和启发式算法来解决任务负载优化问题. 经典的 Min-Min 与 Max-Min 算法[6]通过优先安排计算量过小或过大的任务方式缩短总任务完成时间,但是这类算法的负载均衡较差. Wang 等[7]利用动态规划思想将任务与服务器的匹配作为多阶段决策组合,通过优化任务分配方案来减少任务的完成时间,但是该算法导致了大量开销. 目前,研究学者大多数采用启发式算法来解决云计算任务调度这一多项式复杂程度的非确定性问题(Non-deterministic Polynomial, NP 问题). 启发式算法包括蚁群算法[8]、粒子群算法[9]和乌鸦搜索算法[10]以及相关的改进融合算法. 这类方法具备操作简单、鲁棒性强和扩充性好的特点,能够有效地缩短任务完成时间,解决资源负载均衡问题. 萤火虫算法是根据萤火虫觅食和求偶行为提出的一种启发式算法,具有参数调整少、寻优能力强和简单易实现的特点,广泛应用于多个领域. Ahmed 等[11]通过引入萤火虫算法对云计算中的子任务进行调度,实现了最短的延迟时间. 李成辉等[12]提出了一种改进萤火虫算法的任务调度方法,通过真实物理反弹理论控制搜索区域的萤火虫,提高全局最优概率. 但是上述方法采用的萤火虫算法收敛速度慢,寻优精度不高,容易陷入局部最优,因此在任务调度过程中效果不佳.

① 收稿日期:2020-03-20
基金项目:江苏省高校自然科学基金项目(18KJB480001);全国高等院校计算机基础教育研究会 2018 年重点立项课题(GZYZD2018014);江苏省高等学校自然科学研究面上项目(19KJB520023).
作者简介:朱利华,硕士,副教授,主要从事计算机应用研究.

针对多个云服务器虚拟机之间由于任务调度效率低引起的负载不平衡问题，本文提出了一种改进的萤火虫优化算法，可以在大规模场景中实现虚拟机之间的动态负载均衡，能够在云服务器池中搜索并找到最佳和最近的云服务器，有效解决服务器中的负载不均衡问题，降低用户请求的响应时间.

1 标准萤火虫算法

萤火虫算法(Firefly Algorithm，FA)是一种群智能优化算法[13]，该算法根据自然界中萤火虫个体发光确定自身位置和吸引同伴的启发而提出. 在提出随机优化萤火虫算法时假定：单个萤火虫对其他所有萤火虫都存在吸引力，不存在同性排斥的现象；萤火虫吸引力的大小与自身的亮度成正比，与萤火虫间的相对距离成反比，亮度高的萤火虫吸引力大，会引诱低亮度萤火虫向其靠拢.

萤火虫算法中的两个关键要素是亮度和吸引度. 萤火虫的亮度和吸引度随着萤火虫 i 与萤火虫 j 之间的距离变化而变化. 因此，距离萤火虫 r 处的相对亮度可表示为

$$I = I_0 \times \exp(-\gamma r^k),\ k \geqslant 1 \tag{1}$$

其中，I_0 表示萤火虫自身最大亮度，γ 是光吸收因子，k 是指数因子，一般取 2.

吸引度 β 可被定义为

$$\beta = \beta_0 \times \exp(-\gamma r_{ij}^k) \tag{2}$$

其中，β_0 表示最大吸引度，r_{ij} 表示萤火虫 i，j 之间的欧式距离.

萤火虫 i 向更加明亮的萤火虫 j 移动时，移动位置可以表示为

$$s_i = s_i + \beta(s_j - s_i) + \alpha\xi_i \tag{3}$$

其中，s_i，s_j 表示萤火虫 i，j 的解空间位置，α 表示移动步长，ξ_i 是萤火虫 i 的随机因子，服从[0，1]区间的均匀分布. 式(3)右侧第一项表示萤火虫的当前位置，第二项表示受其他萤火虫吸引而产生的位置变化量，该项体现了算法的全局寻优能力，第三项表示萤火虫在局部的随机搜索移动量，体现了算法的局部寻优能力.

2 基于改进萤火虫算法的任务调度

标准萤火虫算法是以个体的亮度作为移动依据，向单个更亮萤火虫移动的策略保证了算法可以收敛至一个小的区域. 但是，这种策略收敛速度慢，寻优精度不高，而且容易陷入局部最优的困境. 因此，本文在标准萤火虫算法的基础上提出了改进算法，通过优化资源搜索路径，提高云计算任务完成速度，达到缩短任务完成总时间和保持云服务器之间负载动态平衡的目的.

2.1 改进的萤火虫算法

假定将萤火虫中的个体分为两类：产生最强烈亮度的萤火虫定义为占优萤火虫，发光较少的萤火虫定义为弱萤火虫. 则萤火虫群体 F 中存在 m 个弱萤火虫$\{f_i \mid i = 1, 2, \cdots, m\}$ 和 n 个占优萤火虫$\{Df_j \mid j = 1, 2, \cdots, n\}$. 占优萤火虫通过发出强烈的亮光吸引其他弱萤火虫. 在一个特定的区域，所有的弱萤火虫都会被其中的占优萤火虫吸引，从而产生更大的亮度值(Brightness Value，BV)，进而吸引更远区域的弱萤火虫. 假设弱萤火虫以两种搜索模式飞向占优萤火虫：① 从较低亮度处飞离：该值由最后搜索边界(Last Search Border，LSB) 表示. ② 朝亮度高的方向飞行：该值由新搜索边界(new search border，NSB) 表示.

萤火虫通过一条最佳路径寻找伙伴，并采用最短飞行距离、最小能量消耗的思路来优化占优萤火虫群的平衡问题，避免长距离飞行时的资源浪费. 欧几里德距离计算法适用于萤火虫飞行路径的搜索.

通过上述分析，占优萤火虫算法是指通过建立两种搜索路径，使得弱萤火虫以消耗尾部能量最少的方式找到占优萤火虫的方法. 基于这种认知，本文将占优萤火虫搜索算法应用于云计算环境中，解决云服务器中各虚拟机(Virtual Machine，VM) 之间的负载不均衡问题，降低迁移时间和云数据中心之间的响应时间.

2.2 负载均衡优化模型

云服务提供商为最终用户提供最佳的低能耗策略，以便使用有效的云服务水平协议(Service Level A-

greement，SLA)访问云中的 IaaS 服务. 每个云用户在与相邻云服务器通信时应具有 SLA，以确保灵活且可靠地共享文件.

在云计算中，云用户不断向可用的云服务器资源发送请求. 请求映射到预期资源是一个繁琐的过程，用户请求和云计算资源不是一一对应的映射关系. 在本文所提出的技术中，引入虚拟机的云服务器池概念用以将每个用户请求映射到云服务器. 云用户请求可以通过虚拟机随机地提供给云中的任何云服务器. 云服务器选择依赖于基本的云管理策略，这些策略取决于每个云服务器上的实际负载. 任何非抢占式系统都采用了调度策略，如 Round Robin 策略和 First-Come-First-Served 策略[14]. 通过这些策略发现每个云服务器虚拟机(Cloud Server Virtual Machine，CSVM)上的负载都不同. 为了使每个 CSVM 负载平衡，需要动态负载平衡策略.

通过减少云服务器的完成时间，使得每个云服务器和整个云负载平衡的性能都获得提高，同时增强了用户满意度. 假定在云计算系统中，存在 m 个用户提交任务 $D=\{D_1, D_2, \cdots, D_m\}$ 和虚拟机 $VM=\{VM_1, VM_2, \cdots VM_m\}$，$n$ 个云服务器 $CS=\{CS_1, CS_2, \cdots, CS_n\}$，则云计算资源优化问题可以表述为

$$S=\{D, VM, CS, M_{t,vm}, M_{vm,cs}\} \tag{4}$$

其中，$M_{t,vm}$ 代表任务与虚拟机之间的映射关系，$M_{vm,cs}$ 代表虚拟机和云服务器间的映射关系.

假设任务 T_i 被映射到虚拟机 VM_j 上，而 VM_j 的任务通过调度在 CS_k 云服务器上执行，则任务 T_i 在服务器 CS_k 执行的最早完成时间为

$$T_{Last}(T_iM_{t,vm}, CS_k)=ST(CS_k)+ET(T_iM_{t,vm}, CS_k) \tag{5}$$

式(8)中，$ST(CS_k)$ 表示云服务器 CS_k 开始执行任务的时间，$ET(T_iM_{t,vm}, CS_k)$ 表示任务 T_i 经过虚拟机上传至服务器 CS_k 后的预期执行时间.

云服务器 CS_k 上所有任务完成的总时间为

$$T_{sum}(CS_k)=\sum_{i=1}^{m} c_{ik}T_{Last}(T_iM_{t,vm}, CS_k) \tag{6}$$

其中，$c_{ik}=1$ 代表任务 D_i 被调度到云服务器 CS_k 执行，$c_{ik}=0$ 代表任务 T_i 未被调度到云服务器 CS_k 执行，即

$$c_{ik}=\begin{cases}1 & T_iM_{t,vm}=CS_k\\ 0 & T_iM_{t,vm}\neq CS_k\end{cases} \tag{7}$$

全部任务 $D=\{D_1, D_2, \cdots, D_m\}$ 总的完成时间可以表示为

$$T\_total=\sum_{k=1}^{n} T_{sum}(CS_k) \tag{8}$$

云计算资源负载平衡的优化问题就是尽可能降低用户提交任务的完成时间. 本文的优化目标函数定义为

$$Goal(T\_total)=\min\sum_{k=1}^{n} T_{sum}(CS_k) \tag{9}$$

改进的萤火虫算法可以应用于云计算资源负载平衡策略. 在一群萤火虫中，总会有几个占优萤火虫和许多弱萤火虫. 该方法假设占优萤火虫代表云服务器，而弱萤火虫代表云用户. 当云服务器占用大量负载(用户请求)时，就需要以这种方式进行平衡，以便将用户请求传输到其他某个云服务器上，保证尽快完成任务. 如果萤火虫在搜索期间发现占优萤火虫已经被许多其他弱萤火虫重复占据，那么则会通过将过多的弱萤火虫传递给下一个占优萤火虫来平衡负载. 根据此算法，当云用户请求增加到云服务器阈值时，用户将自动转移到下一个云服务器. 此外，弱萤火虫朝占优萤火虫飞行的路径表示附近云服务器可提供的动态负载均衡.

改进萤火虫算法的设计是基于云环境中 VM 之间的动态负载均衡策略. 对于云负载平衡策略之类的大规模场景，这种改进的萤火虫算法能够在云服务器池中搜索和找出最佳、最近的云服务器，从而实现多个云服务器 VM 之间的最佳负载平衡. 这种设计旨在更简单，更有效地解决云计算环境中复杂的任务负载平衡问题. 云服务器的路径搜索可以通过应用本文提出的算法来进行优化，具体流程如表 1 伪码所示.

表1 改进的萤火虫算法

程序：改进的萤火虫算法
开始
萤火虫明亮值 $BV=1$
云用户请求 $=0$
初始化参数
while $BV <$ 亮度阈值 *do*
当前负载 $=$ 云服务器负载信息；
在当前负载中加入萤火虫群；
更新本地负载表；
if 当前负载 $=0$
break；
elseif 随机向量 $<$ 占优萤火虫 *then*
目标服务器 CS_{next} $=$ 随机选择最近占优云服务器；
else
CS_{next} $=$ 选择指定占优云服务器；
end if
占优云服务器 $=$ 占优云服务器-过载云服务器；
云用户请求 $=$ 云用户请求 $+1$；
$BV=BV+1$；
飞向目标服务器 CS_{next}；
end while
发送 *VM* 到云服务器；
end

3 实验结果与分析

为了验证本文提出的改进萤火虫算法的有效性，使用CloudSim仿真软件进行测试实验，并在相同条件下与蚁群优化算法(Ant Colony Optimization，ACO)[8]、乌鸦搜索算法(Crow Search Algorithm，CSA)[10]及萤火虫算法(Firefly Algorithm，FA)[11]等现有的负载平衡方法对比. 本文仿真实验的运行环境为Win10系统下Intel(R)Core(TM) i7-7700HQ处理器.

首先，选择100～1 000个任务量在云计算系统上进行运行，资源数量为60. 通过仿真计算得到不同算法的资源优化方案完成所有任务的时间变化如图4所示. 从图1中可以看出，当任务数量较少时，用户间发生资源竞争和资源冲突的概率较小，不同算法均可以达到云计算资源平衡较优的状态. 因此，不同算法间的性能差别不大.

当选取5 000～40 000个任务，云计算资源数同样为60时，图2给出了不同算法的任务完成时间. 从图2中可以看出，随着任务数量不断增加，任务之间的资源竞争加剧，任务间的冲突概率上升，所有算法完成任务的时间也在逐渐增多. 从完成任务的时间来对比，本文提出的算法在任务完成时间方面要远远少于其他算法.

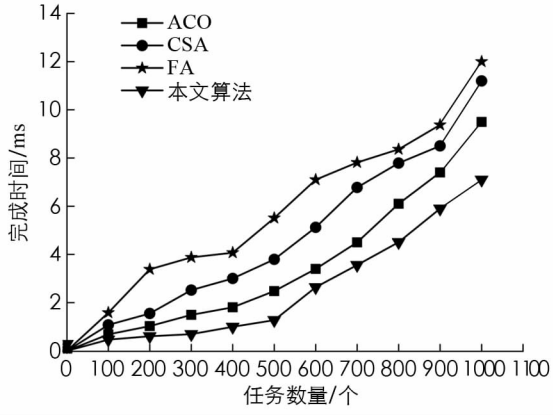

图1 小规模任务量时的任务完成时间对比

当固定任务数量为60 000时，不同云计算资源数量的完成任务时间如图3所示. 由图3可见，随着云计算资源数量不断增加，任务之间的资源竞争降低，使得每个任务均可以在较短时间内得到响应，加快了

任务完成时间. 并且，本文提出的改进萤火虫算法明显优于其他算法.

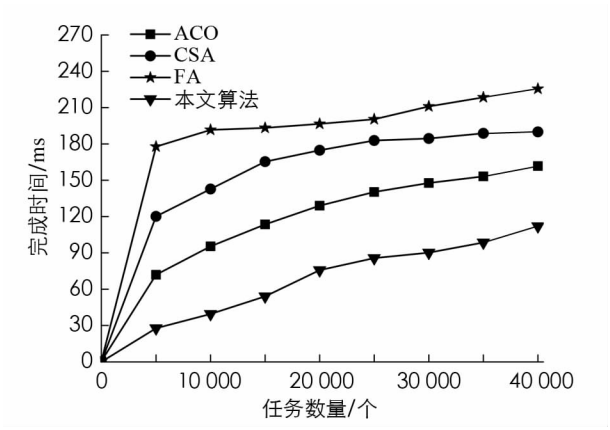

图 2　大规模任务量时的任务完成时间对比

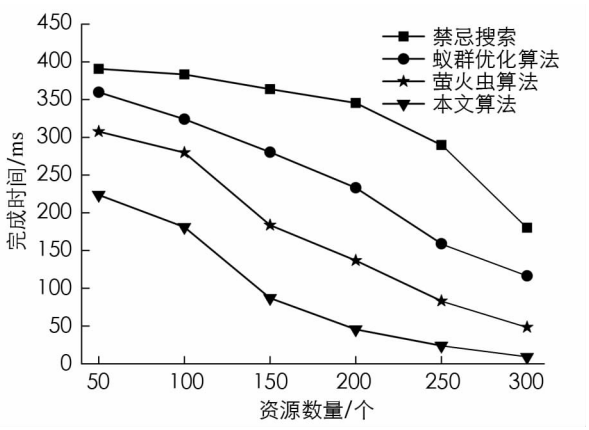

图 3　不同资源数量时的任务完成时间对比

总结上述实验结果可知，基于改进萤火虫算法的任务调度算法可以克服传统萤火虫算法的不足，有效提高精确度，在云计算环境中找到全局最优的负载均衡方案，提高云计算任务的响应时间，达到缩短任务完成总时间的目的. 因此，本文算法能够较好地应用于云计算资源负载均衡调度.

4　结　　语

本文提出了一种基于改进萤火虫算法的虚拟机任务调度策略，用于改善当前云计算中任务调度效率低的问题，提高云服务器间负载均衡的性能. 该算法以缩短传输路径、优化云计算资源负载平衡为出发点，通过改进的萤火虫算法优化资源搜索路径，减少云计算任务的响应时间，达到缩短任务完成总时间的目的，最终结果使得云计算资源利用率得以提高，云计算资源负载平衡得以优化. 仿真实验结果表明，本文算法可以有效地对云计算资源进行调度，任务完成时间明显优于其他算法.

参考文献：

[1] PANDA S K, JANA P K. An Energy-efficient Task Scheduling Algorithm for Heterogeneous Cloud Computing Systems [J]. Cluster Computing, 2019, 22(2): 509-527.

[2] ELAZIZ M A, XIONG S W, JAYASENA K P N, et al. Task Scheduling in Cloud Computing Based on Hybrid Moth Search Algorithm and Differential Evolution [J]. Knowledge-Based Systems, 2019, 169: 39-52.

[3] DE MATOS J G, DE M MARQUES C K, LIBERALINO C H P. Genetic and Static Algorithm for Task Scheduling in Cloud Computing [J]. International Journal of Cloud Computing, 2019, 8(1): 1.

[4] 王　磊，赵晓永. 基于区块链机制的云计算环境下服务组合策略的研究 [J]. 计算机应用研究，2019，36(1)：81-86，90.

[5] ARUL XAVIER V M, ANNADURAI S. Chaotic Social Spider Algorithm for Load Balance Aware Task Scheduling in Cloud Computing [J]. Cluster Computing, 2019, 22(1): 287-297.

[6] LIANG B, DONG X S, WANG Y F, et al. A Low-power Task Scheduling Algorithm for Heterogeneous Cloud Computing [J]. The Journal of Supercomputing, 2020, 76(9): 7290-7314.

[7] WANG P P, DONG K K, LIU H R, et al. Research on Task Allocation and Resource Scheduling Method in Cloud Environment [M]//Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2019.

[8] SELVAKUMAR A, GUNASEKARAN G. A Novel Approach of Load Balancing and Task Scheduling Using Ant Colony Optimization Algorithm [J]. International Journal of Software Innovation, 2019, 7(2): 9-20.

[9] KUMAR A M S, PARTHIBAN K, SHANKAR S S. An Efficient Task Scheduling in a Cloud Computing Environment Using Hybrid Genetic Algorithm-Particle Swarm Optimization (GA-PSO) algorithm [C]//2019 International Conference on Intelligent Sustainable Systems (ICISS). Tokyo: IEEE, 2019.

[10] PRASANNA KUMAR K R, KOUSALYA K. Amelioration of Task Scheduling in Cloud Computing Using Crow Search Algorithm [J]. Neural Computing and Applications, 2020, 32(10): 5901-5907.

[11] AHMED S I, IBRAHIM A O, ABDUTTALIB M S, et al. Task Scheduling for Cloud Computing Based on Firefly Algo-

rithm [J]. Journal of Physics: Conference Series, 2019, 1294: 042004.
[12] 李成辉, 李仁旺, 杨强光, 等. 基于改进萤火虫算法的云计算任务调度算法 [J]. 浙江理工大学学报(自然科学版), 2019, 36(3): 354-359.
[13] LANGARI R K, SARDAR S, AMIN MOUSAVI S A, et al. Combined Fuzzy Clustering and Firefly Algorithm for Privacy Preserving in Social Networks [J]. Expert Systems With Applications, 2020, 141: 112968.
[14] MASHUQUR RAHMAN MAZUMDER A K M, ASLAM UDDIN K M, ARBE N, et al. Dynamic Task Scheduling Algorithms in Cloud Computing [C]//2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA). Coimbatore: IEEE, 2019.

Cloud Computing Task Scheduling Strategy Based on Improved Firefly Algorithm

ZHU Li-hua[1], ZHU Ling-ling[2]

1. *School of Software and Big Data, Changzhou College of Information Technology, Changzhou Jiangsu 213164, China,*
2. *School of Information Science and Technology, Nantong University, Nantong Jiangsu 226200, China*

Abstract: Aiming at the problem of imbalanced resource utilization caused by low task scheduling efficiency in cloud computing, a new cloud computing task scheduling strategy based on improved firefly algorithm has beenproposed. This strategy firstly constructs the constraints of the optimization of cloud computing resource load balance, and takes the minimum user task completion time as the objective function of resource optimization; secondly, optimizes the resource search path through the improved firefly algorithm, optimizes the task load balance among multiple virtual machines in the cloud server, and shortens the user task completion by improving the response efficiency of the cloud server The purpose of the total time. Experimental results show that compared with other algorithms, the proposed strategy has obvious advantages in the completion time of cloud computing tasks, can effectively solve the problem of load imbalance in the server, and improve the response efficiency of user requests.

Key words: cloud computing; task scheduling; firefly algorithm; load balance

责任编辑 夏 娟