基于改进蚁群算法的自适应云资源调度模型研究

聂清彬[1*]，潘峰[1]，吴嘉诚[1]，曹耀钦[2]
[1]西南交通大学希望学院，四川 成都 610400；
[2]重庆工程学院，重庆 400065

摘要 对于传统蚁群算法用于云计算资源分配和调度问题过程中存在的不足，提出了一种可以提高负载均衡度、缩短任务执行时间、降低任务执行成本的改进自适应蚁群算法，改进算法以能够基于用户提交的任务求解出执行时间较短、费用较低，负载率均衡的分配方案为目标，通过 CloudSim 平台对传统蚁群算法、最新的 AC-SFL 算法、改进自适应蚁群算法进行仿真实验对比。实验数据表明，改进后的自适应蚁群算法能够快速找出最优的云计算资源调度问题的解决方案，缩短了任务完成时间，降低了执行费用，保持了整个云系统中心的负载均衡。

关键词 光通信；云计算；自适应；蚁群算法；任务调度
中图分类号 TP393 **文献标志码** A **doi**：10.3788/LOP57.010603

Adaptive Cloud Resource Scheduling Model Based on Improved Ant Colony Algorithm

Nie Qingbin[1*], Pan Feng[1], Wu Jiacheng[1], Cao Yaoqin[2]
[1] *Southwest Jiaotong University Hope College, Chengdu, Sichuan* 610400, *China*;
[2] *Chongqing Institute of Engineering, Chongqing* 400065, *China*

Abstract To address the shortcomings of the standard ant colony algorithm in cloud-computing resource allocation and scheduling, this study proposes an adaptive ant colony algorithm to improve load balance and reduce task execution time and costs. The proposed algorithm can solve tasks submitted by users with a short execution time, low cost, and balanced load rate. The traditional ant colony algorithm, the latest AC-SFL algorithm, and the improved adaptive ant colony algorithm are simulated using the CloudSim platform. Experimental results show that, the improved adaptive ant colony algorithm is able to quickly find a solution for the optimal cloud computing resource scheduling, shorten task completion time, reduce execution cost, and maintain the load balance of the entire cloud system center.

Key words optical communications; cloud computing; self-adaption; ant colony algorithm; task scheduling
OCIS codes 060.4250; 060.4251; 060.4258; 060.4254

1 引 言

云计算[1]是在虚拟化技术和网格计算基础上发展起来的一种新兴的计算模式，也是基于海量计算机资源，以按需分配的方式为用户服务的一种商业模式。其因具有超大规模、数据储存安全可靠和软硬件资源进行统一管理等特点，越来越引起人们的关注。由于云计算资源在互联网中涉及的数据非常庞大，因此如何对云计算中资源进行有效的管理和分配，如何对虚拟机资源进行高效的调度成为云计算研究中所要解决的重要问题。

目前，云计算资源调度策略中的大部分任务分配均选用 Google 公司倡导的分布式编程模式 Map/Reduce，该模式分为两个不同的阶段，即 Map

收稿日期：2019-05-14；**修回日期**：2019-06-21；**录用日期**：2019-07-11
基金项目：2018 年教育部第二批产学协作育人项目立项项目(201802002058)、成都市哲学社会科学研究基地成都市交通＋旅游大数据应用技术研究基地项目(2019001,2018022)
* **E-mail**：3398108124@qq.com

(映射)阶段与 Reduce(化简)阶段:在 Map 阶段,将云计算中心的所有任务细分成若干个子任务,继而调度到每个虚拟资源节点上去执行,任务完成后输出结果;在 Reduce 阶段,首先对 Map 阶段得出的结果进行处理,然后将最后的处理结果反馈给客户,在该并行运算编程模型中,整个算法的核心是如何科学地分割子任务来实现任务的有效分配和云资源调度。

云计算资源在任务调度中所产生的问题是近年来兴起的,国内外的学者都从自己的研究中提出了相应的观点。郭琪瑶等[2]提出了将蚁群算法(ACO)和蛙跳算法进行有效融合,从而达到提高系统处理任务的效率以及云计算资源的合理调度;李建锋等[3]采用一种具备双适应度的遗传算法(DFGA),其算法能有效减少完成任务的平均时间,与自适应算法(AGA)相比,DFGA 的性能明显优于自适应算法(AGA);黄俊等[4]提出了通过加入虚拟机实时状态,更加精确地表达了云计算任务的分配问题,从而缩短了完成任务时间,加快了算法收敛速度;吕燕兵等[5]针对云计算资源分配不均衡和资源利用率不高的问题,提出了基于降低云计算中心负载均衡的改进蚁群算法模型,有效地改善了算法负载的均衡性;魏赟等[6]提出一种既能增强任务并行执行能力的同时又能保持任务串行关系的调度策略——DSFACO 算法。DSFACO 算法将云计算中所需执行的任务分解成若干个子任务,然后根据任务执行顺序放置到优先级不等的调度队列中,针对在同一个调度队列的若干任务,最大限度地将任务被执行的时间缩短;并且在每个任务的延迟时间内也对任务进行调度,从而提高了虚拟机资源的运行效率,使得整个资源调度过程更加公平。该算法使用户提交到云计算中心的任务得到最优的分配。

这些相关算法都在一定程度上解决了传统的云资源调度过程中存在的问题,但云资源调度是一种多用户、多任务、实时并发执行过程,庞大的资源随着任务分配而进行改变。所以,为了缩短云计算中任务的执行时间,有效地提高云计算中虚拟机的利用率,本文将针对传统蚁群算法中存在的缺陷,提出一种改进后的自适应云资源调度算法。

2 标准的蚁群优化算法及资源节点的选择

蚁群算法[7-8]是由 Marco Dorigo 于 1992 年在他的博士论文中所阐述的一种在图中寻找最优路径的算法,是一种模拟自然界蚁群在觅食过程中寻找路径的群体行为人工智能优化算法,算法用来模拟蚂蚁群体去寻觅食物的整个流程,蚂蚁在觅食过程中,在其经过的每一条路径上都会遗留下一种分泌物,叫信息素(pheromone)。因为蚂蚁在选取路径时会更偏好于选取浓度更高的路径,所以容易出现大部分蚂蚁选择同一条信息素浓度较强路径的情况,大量蚂蚁选取浓度更高路径的行为实际上形成了对路径上信息素的整体反馈现象,即选择某一条路径的蚂蚁越多,后面的蚂蚁继续选择该路径的可能性就更大,以此达到路径最短的目的。该算法将蚂蚁寻求最短路径的方法转变为数学问题,具备高求解精度的特点,而且易与其他方法结合,可应用于其他组合优化问题,如旅行商问题、指派问题、车辆路由问题、图着色问题和网络资源调度问题等,能够在海量的解空间中最大程度地获取最优解。尤其在解决组合优化问题方面,通过结合路径上信息素的局部更新与全局更新,计算出下一条路径的状态转移率的值,进而得到全局最优解。

在云计算[9-10]资源环境中,虚拟机的数量设为 $v=\{v_1,v_2,\cdots,v_m\}$,虚拟机表示为 $v=\{v_1,v_2,\cdots,v_m\}$,被云系统划分的子任务的个数为 n,子任务表示为 $T=\{t_1,t_2,\cdots,t_n\}$,将这些子任务提交到虚拟机上去执行,并且在每个时刻,保证每个子任务只在一台虚拟机上运行,通过标准蚁群算法可计算得到任务 t_i 选择某一台虚拟机 v_j 的概率值为

$$p_{ij}^{(k)}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum\limits_{k\in A_{\text{allowed\_}k}}[\tau_{ik}(t)]^{\alpha}[\eta_{ik}(t)]^{\beta}}, & j\in A_{\text{allowed}}\\ 0, & j\notin A_{\text{allowed}}\end{cases},\tag{1}$$

式中:i 和 j 表示路径节点;k 表示第 k 只蚂蚁;$\eta_{ij}(t)$ 表示在 t 时刻路径(i,j)上的启发信息;$\tau_{ij}(t)$ 表示在 t 时刻路径(i,j)上的信息素浓度;β 为信息素启发因子,表示路径上所剩下的信息素浓度对以后任务选择该虚拟机的影响程度;α 为期望启发因子,表示蚂蚁在任务分配过程中,启发信息对任务在选择虚拟机时所起到的相对重要程度;$A_{\text{allowed\_}k}$ 表示第 k 只蚂蚁能够访问的所有下一个路径的集合,是属于禁忌表以外的还没经过的城市。算法迭代次数为 N_{num},最多的迭代次数为 $N_{\max}$。

为了防止某一条路径上的信息素减少或者增加

过多，采用信息素的挥发措施进行约束，以便更有利发现最优的解。每次每只蚂蚁走过后都立刻更新路径上的信息素，更新公式为

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t), \quad (2)$$

式中：$\Delta\tau_{ij}(t)$为 t 时刻路径(i,j)信息素增量；$\Delta\tau_{ij}^{(k)}(t)$为 t 时刻蚂蚁 k 在路径节点 i 到 j 信息素的增量；ρ 表示信息素挥发因子，$\rho\in[0,1)$。

传统蚁群算法[11]优点较多，如算法性能较强，分布式计算实现相对容易，并且容易和其他算法进行融合成为新的算法等。这些优点使得蚁群算法不仅可以很好地运用到实际生活过程中，如物流信息调度、旅行商(TSP)问题等领域，而且也能被很好地应用在资源调度[12]、分配和优化等方面。但是，该算法的随机性很大，因此在解决大规模优化问题时很容易陷入局部最优，导致搜索停滞现象，从而错过了全局最优解，并且该算法的收敛速度相对较慢，在解决云环境中资源管理、分配和调度这种具有大规模性和动态性的任务中，传统蚁群算法寻求最优解的速率和效率就会变得相对较低，阻碍它在云计算资源调度和分配问题中的进一步发展。

3 改进自适应蚁群算法调度

3.1 云计算资源调度

通过对云计算资源调度[13]过程进行分析，可将其描述为：云计算机资源环境中，将云系统划分的 n 个子任务分派至 m 个虚拟机上运行$(m<n)$，并且每个子任务只能选择在一个虚拟资源节点上运行，虚拟机和任务之间的对应分配关系可以用矩阵 $\boldsymbol{H}$ 来表示，表达式为

$$\boldsymbol{H}=\begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2n} \\ \vdots & \vdots & & \vdots \\ h_{m1} & h_{m2} & \cdots & h_{mn} \end{pmatrix}, \quad (3)$$

式中：h_{ij}表示子任务 t_i 与虚拟机 v_j 之间的对应分配关系。云计算机资源调度的算法，就是将待执行的子任务合理地分配到虚拟节点上，在同等条件下，执行算法所执行的分配方案应该尽可能地让整个云系统的负载[14]始终保持均衡，云计算系统执行所需成本相对较低，同时执行任务的时间相对较短。

3.2 时间成本自适应

任务 t_i 在虚拟资源节点 v_j 上的运行时间表示为 T_{ij}，每个任务所需总的运行时间 T_{ij} 是该任务等待虚拟机所产生的延迟时间 W_{ij} 加上该任务在虚拟机上的实际执行时间 E_{ij}，表示为

$$T_{ij}=W_{ij}+E_{ij}, \quad (4)$$

式中：E_{ij}表示任务 t_i 在虚拟机 v_j 上执行时间；W_{ij}表示任务在被执行之前需要等待虚拟机的时间。

E_{ij}表示式为

$$E_{ij}=\frac{T_{\text{tasksize}_i}}{C_{\text{p\_}j}}, \quad (5)$$

式中：T_{tasksize_i}表示任务 t_i 的任务量的大小；$C_{\text{p_}j}$表示虚拟节点 v_j 的运算能力，也代表了该虚拟机的性能，表示式为

$$C_{\text{p\_}j}=W_{\text{vm\_wh}_j}+V_{\text{vm\_ct}_j}-V_{\text{vm\_fa}_j}, \quad (6)$$

其中 $W_{\text{vm_wh}_j}$表示虚拟机 v_j 网络带宽，$V_{\text{vm_ct}_j}$表示虚拟机 v_j 的处理器数量，$V_{\text{vm_fa}j}$表示虚拟机资源节点的故障率。

因为在云计算中，虚拟机执行任务都是并发执行的，所以整个云计算系统将所有任务执行完成的时间也就是所有 T_{ij} 中的最大值，用 $T_{\max}$表示为

$$T_{\max}=\max(T_{ij})。 \quad (7)$$

虚拟资源节点 v_j 在单位时间执行任务消费成本为 $P_{\text{pe\_}v_j}$。执行任务的消耗成本是由该虚拟机上执行任务单位时间的消费成本 $P_{\text{pe\_}v_j}$ 值和该任务的执行时间 E_{ij} 组成，那么分配方案 S 完成任务所需总成本为

$$S_{\text{expenses}}=\sum_{j=1}^{m}P_{\text{pe\_}v_j}\times E_{ij}。 \quad (8)$$

$E_{\max}$表示在计算能力最差的虚拟资源节点上运行客户提交任务所需时间，表达式为

$$E_{\max}=\frac{\sum_{i=1}^{n}T_{\text{tasksize}_i}}{m\times\min(p_j)}, \quad (9)$$

用 $E_{\min}$表示在计算能力最佳的虚拟资源节点上运行客户提交任务所需时间，表达式为

$$E_{\min}=\frac{\sum_{i=1}^{n}T_{\text{tasksize}_i}}{m\times\max(p_j)}, \quad (10)$$

式中：$\min(p_j)$代表最差性能虚拟资源节点运算能力；$\max(p_j)$代表最优性能虚拟资源节点运算能力；m 表示虚拟资源节点数量。

因此时间自适应函数 $t(S)$为

$$t(S)=\frac{t(S)-t_{\min}}{t_{\max}-t_{\min}}, \quad (11)$$

$t(S)$值越小，表示任务执行时间越短。

$$E_{\text{expenses}_{\min}}=E_{\min}\times\min[p(v_j)], \quad (12)$$

$$E_{\text{expenses}_{\max}}=E_{\max}\times\min[p(v_j)],\tag{13}$$

式中：$E_{\text{expenses}_{\max}}$表示在计算能力 $p(v_j)$最佳的虚拟资源节点上运行客户提交任务所需费用；$E_{\text{execution}_{\max}}$表示在计算能力 $p(v_j)$最差的虚拟资源节点上运行客户提交任务所需费用。因此费用自适应函数 $c(S)$对应的表达式为

$$c(S)=\frac{E_{\text{expenses}}(S)-E_{\min}}{E_{\max}-E_{\min}},\tag{14}$$

$c(S)$的值越小，执行任务的费用也就越低。

通过(11)式和(14)式可以计算出自适应因子 $a(S)$的表达式为

$$a(S)=q\times t(S)+w\times c(S),\tag{15}$$

式中：$w\in[0,1]$作为成本因子；$q\in[0,1]$，为时间因子。有 $q+w=1$；唯有当 $q=0.5$，$w=0.5$ 时，整个云计算系统执行任务的时间相对较短，费用较低。将该自适应因子 $a(S)$用于改进蚁群算法中的信息素更新因子 $\tau_{ij}(t)$。改进以后的信息素更新先进行局部更新，然后进行全局更新[15]：

1) 当某一只蚂蚁搜索完成一条路径时，即任务选择了对应的虚拟资源节点以后，会对路径上的虚拟资源节点进行局部信息素更新，这时 $\Delta\tau_{ij}$的表达式为

$$\Delta\tau_{ij}=\frac{U_1}{a(S_{ijk})},\tag{16}$$

式中：U_1 表示局部更新设置的常量；$a(S_{ijk})$表示在第 i 只蚂蚁在第 k 次迭代中为用户计算出的分配策略。

2) 当全部蚂蚁路径搜索完毕，计算出本次迭代中最优的路径，最后对全部虚拟资源节点进行全局信息素更新，这时 $\Delta\tau_{ij}$表示为

$$\Delta\tau_{ij}=\frac{U_2}{\min[a(S_{ijk})]},\tag{17}$$

式中：U_2 为全局更新设置的常量；$\min[a(S_{ijk})]$表示在第 k 次迭代中算法为客户计算得到的最佳的分配方案。

3.3 负载均衡自适应

在云计算环境资源调度过程中，由于云虚拟资源节点的性能不同，每个虚拟资源节点的计算能力、网络带宽等不同，云计算系统始终处在一个动态分配的过程中。每个虚拟机资源节点在每个时刻所承担的负载差异较大，一部分虚拟资源节点性能较差，另外一部分的资源节点性能较好，大量的任务容易集中到性能优越的虚拟机上执行，而性能较差虚拟机却很可能处于一个空闲状态。整个云计算系统的负载差异大、不均衡，导致系统的资源利用率低。为了在任务执行时，保持整个系统负载均衡，将负载因子作为任务选择虚拟机的一个重要考量因素。本文利用负载因子改进传统蚁群算法中的启发函数，让更多的负载大的虚拟机执行较少的任务，负载小的虚拟机执行更多的任务，从而使得整个云系统的资源调度更加科学，让系统始终保持在一个负载均衡的运行状态。

在云计算执行任务过程中，虚拟机的负载主要是由其执行的任务量的大小和自身的计算能力所决定的，虚拟机 v_j 的负载公式为

$$J=f_1\times P_{\text{pe}}+f_2\times B_{\text{be}}+f_3\times M_{\text{me}},\tag{18}$$

式中：P_{pe}为虚拟节点 CPU 的占用率；B_{be}为网络带宽的占用率；$f_1+f_2+f_3=1$，分别表示 CPU、带宽、内存所占权重的大小。影响负载的因素有虚拟节点 CPU 的占用率 P_{pe}，网络带宽的占用率 B_{be}，内存的占用率 M_{me}。将 J 用于改进传统蚁群算法中的 η 因子，公式为

$$\eta_{ij}=\frac{1}{J}。\tag{19}$$

根据(19)式可得，每个任务在被执行之前，需要从多个虚拟机中选择一个虚拟机来执行任务，为此，需要提前计算该任务选择每个虚拟机所对应的 η_{ij}值。η_{ij}越小，说明任务选择该虚拟机的 J 值偏大，即负载偏大，选择该虚拟机会使整个云计算系统负载更加不均衡；反之如果所得出的 η_{ij}值越大，则对应的 J 值偏小，即负载偏小，选择该虚拟机执行任务将会促进整个云计算系统的负载均衡。因此该改进算法可以促使任务在一些空闲的虚拟机上执行，经过算法多次迭代以后，改进后的算法最终能够实现云计算中心虚拟机的负载平衡。将负载均衡自适应因子用于改进标准的蚁群算法，改进算法为

$$p_{ij}^{(k)}=\begin{cases}\dfrac{(\tau_{ij})^{\alpha}\left(\dfrac{1}{J}\right)^{\beta}}{\displaystyle\sum_{s\in A_{\text{allowed}_k}}(\tau_{ij})^{\alpha}\left(\dfrac{1}{J}\right)^{\beta}}, & j\in A_{\text{allowed}_k}\\ 0, & \text{otherwise}\end{cases}。\tag{20}$$

通过增加云计算资源调度过程中的负载均衡自适应因子，改进了传统的蚁群算法。改进算法可以最大化调度云计算资源，能够使任务分配到最适合的云虚拟资源节点上，完成了任务与虚拟机的最佳匹配，实现了虚拟资源消费相对较低、执行任务时间

相对较短、保持整个云系统的负载均衡的目的，同时增强了对全局最优解的搜寻能力，让改进后的蚁群优化算法具有对未知路径的探索能力更强，从而打破陷入局部最优解后停滞搜索的状态，增强了求解全局最优解的概率，有效地缩短了对大规模数据处理所需要时间。

4 自适应蚁群算法流程

改进后的蚁群优化算法不管是在蚁群的搜索前期还是后期，都能有效地提高获得全局最优解的效率，并且根据时间成本自适应约束因子和负载均衡自适应因子寻找从起始地到目的地的最短路径，弥补了标准蚁群算法 BACO 对全局搜索能力弱的不足。

Step1 变量初始化：对蚁群算法中的各个参数进行初始化，包括信息启发式因子 α，期望启发因子 β，信息素浓度 τ，启发信息函数 η，蚂蚁个数 m；设置迭代次数 N_{num} 与最大迭代次数 N_{max}。

Step2 随机部署：将 n 只蚂蚁随机部署于起始虚拟机资源节点上。

Step3 计算任务到资源的负载自适应因子 J 作为启发信息，计算任务到虚拟机执行的约束函数 $a(S)$，根据（20）式计算出每个虚拟机被选择的概率，并且利用轮盘赌算法最终确定执行该任务的虚拟机。

Step4 如果当前任务对虚拟机的选择完成，通过（16）式对路径上的虚拟机进行局部信息素更新，如果没完成，跳转到步 Step2。

Step5 如果所有任务对虚拟机的选择完成，根据任务和虚拟机的分配情况，找出最佳的分配方案，通过（17）式对路径上的所有虚拟机进行全局信息素更新。

Step6 判断当前迭代次数是否为最大值，如已满足，结束整个流程，如果没达到最大值，继续执行 Step2。

5 仿真结果与分析

为了方便快捷地验证改进后蚁群算法求解云计算资源分配和调度问题的可行性和有效性，采用云仿真工具 CloudSim 对改进算法进行仿真实验，探究了相关资源的分配和调度过程，通过重写 DatacenterBroker 类，将改进后的自适应大规模数量的蚁群优化算法应用到其上，实现对改进的资源分配和调度算法的模拟仿真。在同等实验条件下，将传统的蚁群算法 BACO、参考文献[2]提出的最新改进算法 AC-SFL 与本文的改进蚁群算法 IACO 进行仿真对比。

仿真设定任务数量从 100 到 500，计算节点数 30 个，同时设置上述改进后蚁群算法所需的参数：蚂蚁总数 30 只，信息启发式因子 $\alpha=1$，期望启发因子 $\beta=1$，信息素强度相同，挥发系数初始值 $\rho=0.2$。采用 CloudSim 分别将三种算法进行多次实验对比，将获取数据利用折线图进行绘制。

从图 1 可以看出，改进的蚁群算法 IACO 在整个云计算中心的负载率基本在 0.4 到 0.5 之间，整个系统负载率较低，系统负载保持均衡，传统的蚁群算法 BACO 的负载很不均衡，负载率在 0.82 到 0.90 之间波动，负载率较高，系统承受压力较大，AC-SFL 虽然在负载均衡度上有一定的改进，但是一直在 0.6 到 0.7 之间波动，整个云系统负载不均衡，实验数据结果表明，改进的蚁群算法 IACO 负载均衡，效果明显，云计算中心的负载始终保持在一个均衡的状态。

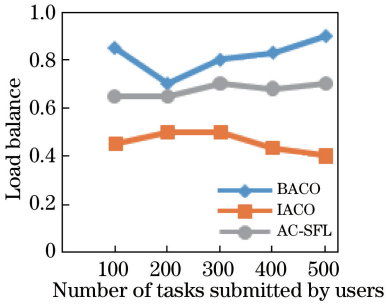

图 1 负载均衡度对比

Fig. 1 Load balance comparison

从图 2 能够看出，当任务量在 100 到 200 之间时，传统的蚁群算法 BACO、AC-SFL 和改进蚁群算法 IACO 所产生费用相差不大，但是随着任务数量的增加，在执行同样数量任务的情况下，传统的蚁群算法 BACO 和 AC-SFL 成本有一定的降低，但远高

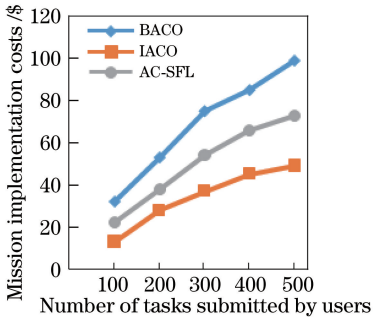

图 2 任务执行费用对比

Fig. 2 Task execution cost comparison

于本文中 IACO 算法的成本，说明 IACO 在降低成本方面性能优越。

从图 3 可知，当任务数量在 100 左右的时候，传统的蚁群算法 BACO、AC-SFL 和改进蚁群算法 IACO 在执行同样数量任务情况下，所需时间都差别不大，但随着任务的不断增加，改进蚁群算法 IACO 执行同样数量任务的时间都远远低于传统的蚁群算法 BACO 和 AC-SFL，说明 IACO 在缩短时间方面性能优越。实验证明，改进后的自适应蚁群算法 IACO 在云计算任务分配策略中，很大程度上缩短了任务执行时间，降低了执行成本，同时降低了负载均衡率，效果显著。

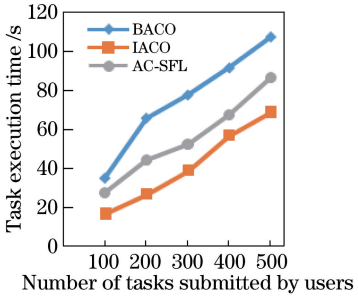

图 3 任务执行时间对比

Fig. 3 Task execution time comparison

6 结 论

本文提出了一种基于改进自适应蚁群算法的云任务分配模型。首先对标准蚁群算法 BACO 和云计算资源调度模型进行分析阐述，然后利用改进的自适应改进蚁群算法 IACO 对云计算任务分配进行求解，最后利用 CloudSim 仿真平台对自适应蚁群算法进行仿真实验。实验表明，改进的自适应蚁群算法明显有效优化了资源调度，缩短了任务执行时间，降低了任务执行成本，提高了云计算资源分配和调度效率，让云计算系统稳定地保持在一个负载均衡状态。

参 考 文 献

[1] Lin W W, Zhu C Y. A scalable distributed scheduling method for large-scale cloud resources[J]. Computer Engineering and Science, 2015, 37(11): 1997-2005.
林伟伟, 朱朝悦. 面向大规模云资源调度的可扩展分布式调度方法[J]. 计算机工程与科学, 2015, 37(11): 1997-2005.

[2] Guo Q Y, Zhu F D. Cloud computing resource scheduling algorithm based on ant colony algorithm and leapfrog algorithm[J]. Bulletin of Science and Technology, 2017, 33(5): 167-170.
郭琪瑶, 朱范德. 基于蚁群算法和蛙跳算法的云计算资源调度算法[J]. 科技通报, 2017, 33(5): 167-170.

[3] Li J F, Peng J. Task scheduling algorithm based on improved genetic algorithm in cloud computing environment[J]. Journal of Computer Applications, 2011, 31(1): 184-186.
李建锋, 彭舰. 云计算环境下基于改进遗传算法的任务调度算法[J]. 计算机应用, 2011, 31(1): 184-186.

[4] Huang J, Wang Q F, Liu Z Q, et al. Cloud task scheduling based on resource state ant colony optimization[J]. Computer Engineering and Design, 2014, 35(9): 3305-3309.
黄俊, 王庆凤, 刘志勤, 等. 基于资源状态蚁群算法的云计算任务分配[J]. 计算机工程与设计, 2014, 35(9): 3305-3309.

[5] Lü Y B, Wang J Y, Wu J M. Research on load-balance resource scheduling algorithm in cloud computing[J]. Journal of Inner Mongolia University of Science and Technology, 2017, 36(2): 181-186.
吕燕兵, 王静宇, 吴金明. 云计算资源负载均衡调度优化算法研究[J]. 内蒙古科技大学学报, 2017, 36(2): 181-186.

[6] Wei Y, Chen Y Y. Cloud computing task scheduling model based on improved ant colony algorithm[J]. Computer Engineering, 2015, 41(2): 12-16.
魏赟, 陈元元. 基于改进蚁群算法的云计算任务调度模型[J]. 计算机工程, 2015, 41(2): 12-16.

[7] Hua X Y, Zheng J, Hu W X. Ant colony optimization algorithm for computing resource allocation based on cloud computing environment[J]. Journal of East China Normal University (Natural Science), 2010(1): 127-134.
华夏渝, 郑骏, 胡文心. 基于云计算环境的蚁群优化计算资源分配算法[J]. 华东师范大学学报(自然科学版), 2010(1): 127-134.

[8] Zhang H R, Chen P H, Xiong J B. Task scheduling algorithm based on simulated annealing ant colony algorithm in cloud computing environment [J]. Journal of Guangdong University of Technology, 2014, 31(3): 77-82.
张浩荣, 陈平华, 熊建斌. 基于蚁群模拟退火算法的云环境任务调度[J]. 广东工业大学学报, 2014, 31(3): 77-82.

[9] Zhang H Q, Zhang X P, Wang H T, et al. Task

scheduling algorithm based on load balancing ant colony optimization in cloud computing [J]. Microelectronics & Computer, 2015, 32(5): 31-35, 40.
张焕青, 张学平, 王海涛, 等. 基于负载均衡蚁群优化算法的云计算任务调度[J]. 微电子学与计算机, 2015, 32(5): 31-35, 40.

[10] Zuo L Y, Zuo L F. Cloud computing scheduling optimization algorithm based on reservation category [J]. Computer Engineering and Design, 2012, 33(4): 1357-1361.
左利云, 左利锋. 云计算中基于预先分类的调度优化算法[J]. 计算机工程与设计, 2012, 33(4): 1357-1361.

[11] Agarwal M, Srivastava G M S. A genetic algorithm inspired task scheduling in cloud computing [C] // 2016 International Conference on Computing, Communication and Automation (ICCCA), April 29-30, 2016, Greater Noida, India. New York: IEEE, 2016: 364-367.

[12] Wang T T, Liu Z B, Chen Y, et al. Load balancing task scheduling based on genetic algorithm in cloud computing [C] // 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing, August 24-27, 2014, Dalian, China. New York: IEEE, 2014: 146-152.

[13] Sheng X D, Li Q. Template-based genetic algorithm for QoS-aware task scheduling in cloud computing[C]// 2016 International Conference on Advanced Cloud and Big Data (CBD), August 13-16, 2016, Chengdu, China. New York: IEEE, 2016: 25-30.

[14] Song W Z, Yang B, Zhao X H, et al. A fast and scalable supervised topic model using stochastic variational inference and MapReduce[C]//2016 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC), September 23-25, 2016, Beijing, China. New York: IEEE, 2016: 94-98.

[15] Chen X, Song W F, Li Z G. Research of resource scheduling based on ACA-GA in the cloud computing [J]. International Journal of Grid and Distributed Computing, 2016, 9(6): 1-12.