·云计算专题· 文章编号:1000-3428(2018)08-0014-05 文献标志码:A 中图分类号:TP391

云计算中任务调度优化策略的研究

全 力,傅 明

(长沙理工大学 计算机与通信工程学院,长沙 410114)

摘 要: 针对基于蚁群算法的任务调度负载不均衡与收敛速度较慢的问题,提出一种改进的任务调度优化算法。通过赋予权重的方法对蚁群算法的信息素更新规则进行优化,加快求解速度,利用动态更新挥发系数优化算法的综合性能,并在局部信息素的更新过程中,引入虚拟机负载权重系数,从而保证虚拟机的负载均衡。实验结果表明,改进算法的任务调度策略在保证任务得到合理分配的同时,还可以提高收敛速度并缩短总执行时间。

关键词: 蚁群算法;信息素;虚拟机;权重系数;收敛速度

中文引用格式: 全 力,傅 明. 云计算中任务调度优化策略的研究[J]. 计算机工程,2018,44(8):14-18.

英文引用格式: QUAN Li, FU Ming. Research on task scheduling optimization strategy in cloud computing[J]. Computer Engineering,2018,44(8):14-18.

Research on Task Scheduling Optimization Strategy in Cloud Computing

QUAN Li,FU Ming

(School of Computer and Communication Engineering,Changsha University of Science and Technology,Changsha 410114,China)

【Abstract】 Aiming at the problem that task scheduling based on ant colony algorithm has unbalanced load and slow convergence speed,an improved task scheduling optimization algorithm is proposed. The pheromone update rules of the ant colony algorithm are optimized by weighting methods to accelerate the solution speed, and the comprehensive performance of the dynamic update volatilization coefficient optimization algorithm is utilized, and the load weight coefficient of the virtual machine is introduced during the update process of the local pheromone to ensure the load balancing of virtual machines. Experimental results show that the task scheduling strategy of the improved algorithm ensures that the task is reasonably allocated,and at the same time,the convergence speed of the algorithm is improved and the total execution time is shortened.

【Key words】 ant colony algorithm;pheromone;virtual machine;weight coefficient;convergence speed

DOI: 10.19678/j.issn.1000-3428.0049169

0 概述

云计算是互联网计算发展的新革命,相比于分布式计算其具有更突出的优势,它是一种高效率的多功能计算系统[1],大规模的计算资源确保了云服务的高效性。云计算中需要处理的任务被调度分配到在各个计算资源中,用户可以通过云计算按照需求获取计算与信息服务,以及高效的存储服务[2]。任务调度作为云计算的核心技术,在云计算处理任务的过程中,任务调度是不可避免的重要环节之一,因此,优化任务调度机制是强化云计算综合性能的重要方法。

为了更有效地改善云计算的服务性能,有学者针对云计算中的任务调度机制所遇到的问题展开了研究。文献[3]以提高云计算服务质量为基础,通过对蜂群算法的改进,能够在一定程度上提高算法的全局搜索能力,并且能够有效地满足 QoS 需求;文献[4]引入局部搜索算子对蜂群算法进行优化,能够有效地缩短任务的完成时间;文献[5]利用蚁群算法对云计算任务调度系统进行改善,能够较好地在多个方面提高系统性能;文献[6]针对任务调度过程中虚拟机负载的问题,利用蚁群算法较好的反馈机制提出蚁群优化算法,能够较好地改善资源利用率;文献[7]利用遗传算法较好的搜索能力能够有效地改善任务调度的性能,使虚拟机达到负载均衡;文献[8]利用遗传算法前期搜索能力强的优势与蚁群算法的信息反馈的特点,将 2 种算法相结合来弥补各自的问题,该算法在改善收敛速度的同时满足用户所要求的服务质量;文献[9]对遗传算法中的交叉和变异的概率公式进行优化,提高

作者简介: 全 力(1991—),男,硕士研究生,主研方向为云计算、信息安全;傅 明,教授、博士。

收稿日期: 2017-11-02 **修回日期:** 2017-12-05 **E-mail:** 120932797@qq.com

了全局搜索能力;文献[10]针对云计算中任务调度需要多个目标优化的问题,提出了改进的 Memetic 算法,该算法能够较快地找到全局最优解,提高了计算效率。

本文针对蚁群算法收敛速度慢与任务分配不均的问题,结合调度机制的特点对算法中信息素更新方法进行优化,通过对信息素增量赋予权值,改进算法的计算效率,并采用挥发系数动态调整的方法,提高算法的搜索能力,在进行局部信息素更新时,加入衡量虚拟机工作负载的权重系数,保证虚拟机的负载均衡。

1　任务调度模型

云计算在处理任务时,任务需要被合理地分配到各个计算资源中,其任务调度机制可以表示为:通过任务调度策略将需要处理的任务分配到异构的资源上,使全部任务在被执行完成之后所需时间最少并满足负载均衡。本文考虑的问题是将 N 个相互独立、不同大小的任务根据任务调度算法分配到 M 个性能不同的虚拟机进行并行计算,根据虚拟机处理任务时的性能指标得出实验结果。

在云计算调度模型中需要处理的作业为 N 个互不关联的任务,N 个任务集合表示为:$Task = \{t_1, t_2, \cdots, t_n\}$,$MI$ 表示任务的指令长度,其中 MI_i 表示任务 t_i 的指令长度[11]。处理任务的异构资源为 M 个虚拟机,虚拟机集合表示为 $VM = \{v_1, v_2, \cdots, v_m\}$,$MIPS$ 表示虚拟机的执行速度,其中 $MIPS_j$ 表示虚拟机 v_j 的执行速度[12]。

为计算每个任务在不同虚拟机上所需要的处理时间,定义执行时间矩阵为:

$$\boldsymbol{T} = \begin{bmatrix} time_{11} & time_{12} & \cdots & time_{1m} \\ time_{21} & time_{22} & \cdots & time_{2m} \\ \vdots & \vdots & & \vdots \\ tme_{n1} & time_{n2} & \cdots & time_{nm} \end{bmatrix} \tag{1}$$

其中,$time_{ij}$表示表示虚拟机 v_j 处理任务 t_i 所需要的执行时间,并且 $time_{ij} = MI_i / MIPS_j$。

根据虚拟机与任务的匹配关系建立矩阵 $\boldsymbol{x}[i][j]$,表示任务 t_i 是否分配给虚拟机 v_j,定义为:

$$\boldsymbol{x}[i][j] = \begin{cases} 1, task_i \text{ 分配给 } v_j \\ 0, \text{其他} \end{cases} \tag{2}$$

通过计算任务完成的总执行时间,对虚拟机的执行效率进行评估,为了计算总执行时间,定义时间集合 $C = \{c_1, c_2, \cdots, c_m\}$,$c_j$ 表示虚拟机 v_j 完成分配的任务所需要的执行时间,其中 $c_j = \sum_{i=1}^{n} time_{ij} \times x_{ij}$。由于虚拟机在处理任务时采用并行计算[13],因此任务完成的总执行时间表示为:$time = \max_{j \in m}(c_j)$,$m$ 表示虚拟机的 ID 号。

负载均衡度作为衡量虚拟机工作效率的关键指标[14],定义为:$LB = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(c_j - \bar{c})^2}$,其中 $\bar{c} = \sum_{j=1}^{m} c_j / m$ 表示每台虚拟机完成任务的平均执行时间,负载均衡度表示为任务执行时间标准差。

2　云计算任务调度策略

2.1　基于蚁群算法的调度模型

蚁群算法作为一种全局优化算法,一般被用来解决路径优化的问题,该算法具有分布性、随机性、反馈性等特点,在云环境中利用蚁群算法的特点能够有效地处理任务调度机制所遇到的问题。

在应用蚁群算法解决任务调度问题的过程中,首先根据虚拟机对每个任务的处理能力,对信息素 $\tau_{ij}(t)$ 与启发函数 $\eta_{ij}(t)$ 进行初始化,并对任务集合 E 初始化,E_k 表示第 k 只蚂蚁可以选择的任务集合,$E_k = \{t_1, t_2, \cdots, t_n\}$,根据信息素浓度与启发函数计算任务与虚拟机的配对概率 $p_{ij}^k(t)$:

$$p_{ij}^k(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}(t)]^{\beta}}{\sum_{j \in allowed_k} [\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}(t)]^{\beta}}, & j \in allowed_k \\ 0, \text{其他} \end{cases} \tag{3}$$

其中,$allowed_k$ 表示第 k 只蚂蚁可以选择的虚拟机集合,$allowed_k = \{v_1, v_2, \cdots, v_m\}$,$p_{ij}^k(t)$ 表示第 t 次迭代时,蚂蚁 K 将任务 t_i 分配給虚拟机 v_j 的概率。启发函数表达式为:$\eta_{ij}(t) = 1/D_{ij}$,$D_{ij} = 1/time_{ij}$ 表示为虚拟机 v_j 对任务 t_i 的执行时间,执行时间越小启发函数 $\eta_{ij}(t)$ 越大。

在完成一次迭代之后,根据虚拟机与任务的配对情况,对信息素 $\tau_{ij}(t)$ 进行更新,表达式为:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{4}$$

其中,$\tau_{ij}(t)$ 表示在 t 时刻虚拟机 v_j 与任务 t_i 配对的信息量,信息素挥发系数 ρ 表示为信息素的积累程度,目的是防止某些信息素在累加过程中积累过多,提高搜索范围,丰富解的多样性[15]。$\Delta\tau_{ij}(t)$ 表示在本次迭代之后虚拟机 v_j 与任务 t_i 配对的信息素增量,其中:$\Delta\tau_{ij}(t) = \sum_{k=1}^{n} \Delta\tau_{ij}^k(t)$。最优解 F_{best}的表达式为:

$$F_{\text{best}} = \begin{cases} F_a(t), & F_a(t) \leqslant F_{\text{best}} \\ F_{\text{best}}, & \text{其他} \end{cases} \tag{5}$$

其中,$F_a(t) = \min_{k=1}^{n}(time_k(t))$ 表示第 t 次迭代的全局最优解,其物理意义为任务完成的时间,n 表示蚂蚁的数量。

2.2 任务调度算法的优化

蚁群算法在解决任务分配的问题时,对任务的分配采用随机选择的方法,影响了算法的搜索能力,使算法需要更多的迭代才能得到最优分配方案,因此,会出现收敛速度变慢的问题,虽然通过正反馈机制可以强化较优的解,但会导致停滞现象。针对这些问题,本文通过对信息素增量 $\Delta\tau_{ij}(t)$ 赋予权值的方法,改变全局信息素的更新规则,并依据迭代次数的增加更新挥发系数 ρ 的参数值,改善了算法的综合性能。

2.2.1 信息素的更新

为了更好地利用最优解的信息,使算法得到较好的性能,在每次迭代完成之后,将每只蚂蚁所形成的解按照从小到大排序,表示为 $F_1(t)\leqslant F_2(t)\leqslant\cdots\leqslant F_m(t)$,利用解的大小对信息素增量赋予不同的权值,最优解的权值最大。最优解的权值大小为 a,当全部的蚂蚁进行完一次迭代之后,利用赋予信息素增量权值的方法对信息素做全局更新:

$$\tau_{ij}(t+1)=\rho\cdot\tau_{ij}(t)+\sum_{n=1}^{a-1}(a-n)\cdot\Delta\tau_{ij}^{n}(t)+a\cdot\Delta\tau_{ij}^{a}(t) \tag{6}$$

$$\Delta\tau_{ij}^{n}(t)=\begin{cases}Q/F_n(t), & \text{在第 } n \text{ 个解中 } t_i \text{ 分配给 } v_j\\ 0, & \text{其他}\end{cases}$$

其中,Q 表示信息素强度系数,$F_n(t)=\max_{j=1}^{m}(c_j^n(t))$ 表示本次迭代排序大小为 n 的解。

负载均衡是衡量任务调度算法的重要指标,针对负载均衡的问题,在信息素的更新规则中,通过加入局部信息素的更新方法来保障任务调度的负载均衡。局部信息素的更新方法定义为:每完成一次任务与虚拟机的配对之后,将引入局部信息素,并依据虚拟机的运行时间对其进行更新,更新规则为:

$$\tau_{ij}(t)=\rho\tau_{ij}+\varphi_j^k(t)\tau_{ij}^0(t) \tag{7}$$

其中,τ_{ij}^0 为信息素的初值,τ_{ij}^0 的大小表示为:$\tau_{ij}^0=1/time_{ij}$,$\varphi_j^k(t)(\varphi\in(0,1))$ 为衡量虚拟机负载的权重系数:

$$\varphi_j^k(t)=\frac{time_{\max}^k(t)-time_j^k(t)}{time_{\max}^k(t)-time_{\min}^k(t)} \tag{8}$$

其中,$time_{\max}^k(t)$ 表示在第 k 只蚂蚁选择的分配方案中虚拟机完成任务的最大时间,$time_{\min}^k(t)$ 表示最快时间,$time_j^k(t)$ 表示虚拟机 v_j 完成分配的任务所需的时间。如果在分配方案中虚拟机 v_j 没有与任务 t_i 配对,则 τ_{ij} 将不进行局部信息素的更新。

2.2.2 自适应调整策略

信息素挥发系数 ρ 的数值设定是否合理将会影响算法选找最优分配方案的搜索能力与计算效率[16],通过动态地改变 ρ 的值,从而能自适应地调整信息素的大小,因此,可以有效地加强算法前期的搜索能力,丰富解的多样性,并随着迭代次数的增加,最优分配方案的概率得到提高,保证了算法的综合性能,挥发系数 ρ 的调整策略为:

$$\rho_a(t)=1-\ln(t)/\ln(t+c) \tag{9}$$

$$\rho(t)=\begin{cases}\rho_{\min}, & \rho_a(t)<\rho_{\min}\\ \rho_a(t), & \rho_{\min}(t)\leqslant\rho_a(t)\leqslant\rho_{\max}\\ \rho_{\max}, & \rho_a(t)\geqslant\rho_{\max}\end{cases} \tag{10}$$

其中,C 为常数,挥发系数 ρ 的大小限制在 $[\rho_{\min},\rho_{\max}]$,使 ρ 的值不会因为过大或过小,而导致求解效率变慢或者搜索能力变弱,保证了算法的综合性能。

2.3 算法的实现过程

算法实现过程如下:

1)初始化启发函数 $\eta_{ij}(t)$、信息素 $\tau_{ij}(t)$、最大迭代次数 $t_{\max}$、蚂蚁数量 n。

2)蚂蚁 k 从任务集合 E_k 中选择一个任务,并根据选择概率 $p_{ij}^k(t)$ 的大小分配给虚拟机。

3)更新蚂蚁 k 的任务禁忌表 $tabu_k$,如果蚂蚁 k 将任务 t_i 分配给虚拟机 v_j,则将任务 t_i 加入禁忌表 $tabu_k$,更新任务集合 E_k,如果任务集合 E_k 为空,则执行下一步,否则将跳转到第 2 步。

4)更新局部信息素,计算蚂蚁 k 生成的解 $F_k(t)$,$F_k(t)=\max_{j=1}^{m}(c_j^k(t))$ 表示任务完成的总执行时间。如果 $k<n$,则 k++ 并返回第 2 步,否则将执行下一步。

5)全局信息素的更新,根据式(5)对全局最优解进行更新,判断是否为最大迭代次数,如果 $t\geqslant t_{\max}$,则结束算法,否则 t++、$k=1$ 并跳转到第 2 步。

算法伪代码如下:

```
1. initialize η_ij(t),τ_ij(t),t_max,n,t←1;
2. while t≤t_max
3. initialize k←1,tabu_k,E_k;
4. calculate p_ij^k(t);
5. if k≤n
6. while E_k≠ϕ
7. a task is assigned to a vm according top_ij^k(t);
8. update tabu_k,E_k;
9. end while
10. update τ_ij(t) by 式(7);
11. calculate F_k(t);
12. calculate p_ij^k(t);
13. k++;
14. end if
15. update τ_ij(t) by 式(6);
17. update F_best by 式(5);
18. t++;
19. end while
```

3 实验结果与分析

实验选择云计算仿真器 CloudSim 对任务调度实验进行仿真,CloudSim 是一种可扩展、通用的仿真

框架,能够对云计算任务调度实验进行模型建立和仿真模拟。实验对本文设计的改进蚁群算法与 Min-Min 算法、基本蚁群算法进行仿真,并比较 3 种不同算法的实验结果。

1)Min-Min 算法

采用 Min-Min 算法处理任务调度的过程为:通过计算每个虚拟机处理不同任务时所需要的完成时间,选出每个任务所需完成时间的最小值,根据最小值的大小将任务进行排序,按照由小到大的顺序依次分配任务,因为 Min-Min 算法只考虑任务被完成的时间为优化目标,所以会导致任务分配不合理的问题。

2)蚁群算法

在应用蚁群算法解决任务调度问题的过程中,首先根据虚拟机对每个任务的处理能力,计算任务与虚拟机的配对概率,由蚂蚁根据配对概率对任务进行分配,完成一次迭代之后,更新目标解与信息素,在算法完成收敛时得到目标解。由于随机选择的方法与反馈机制的原因,因此会导致收敛速度变慢与早熟现象。

参数设置:需要处理的任务个数分别设为 40、60、80、100 和 120,任务的指令长度为 5 000 MI ~ 50 000 MI,虚拟机数量为 8,执行速度为 1 000 MIPS ~ 2 000 MIPS。根据文献[6,17],并通过多次实验进行测试仿真,对算法参数进行设置,算法参数如表 1 所示。

表 1　任务调度算法参数

算法参数	参数值	算法参数	参数值
α	2	$\rho_{\min}$	0.2
β	2	$\rho_{\max}$	0.8
Q	1	n	20.0

为了验证改进蚁群算法应用于云任务调度时的收敛速度,实验方案设置为:对于不相同的任务个数,分别计算改进蚁群算法与基本蚁群算法 2 种方案,找到最优解时的迭代次数,虚拟机的个数为 8。每组数据为 10 次实验的平均值,实验结果如图 1 所示。

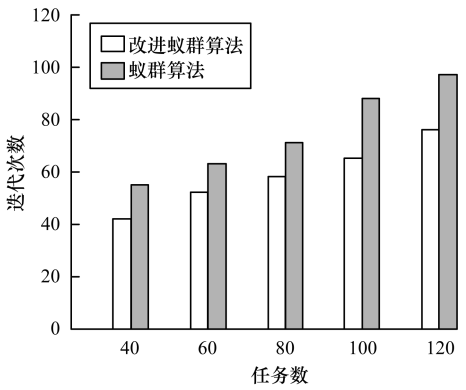

图 1　算法的迭代次数

图 1 的仿真结果表明:改进蚁群算法完成收敛所需的迭代次数低于基本蚁群算法,因为本文通过结合调度机制的特点对信息素的更新规则进行优化,通过对信息素增量 $\Delta\tau_{ij}(t)$ 赋予权值的方法,加快算法的求解速度,并对挥发系数的值进行动态更新,随着迭代次数的增加,提高最优目标解的概率,进一步加快算法的计算效率,收敛速度得到提高,因此改进蚁群算法相比于基本蚁群算法能够更快地找到最优解,算法性能得到较好地改善。

图 2 为在任务个数不相同的情况下,基于 3 种不同调度算法虚拟机完成任务所需要的总执行时间的对比。

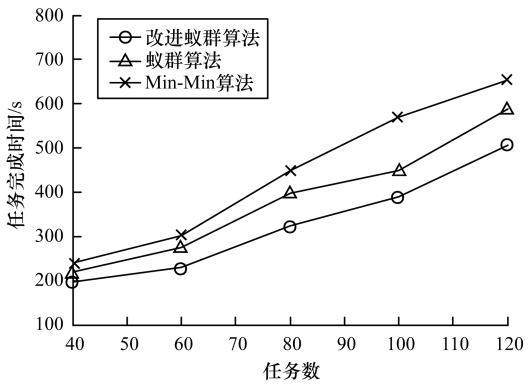

图 2　不同算法的任务完成时间

通过对 3 种算法的对比,改进蚁群算法根据解的大小作为信息素更新时的权值,并依据迭代次数的大小计算挥发系数,使算法的全局搜索能力得到加强,丰富解的多样性,最终解的大小更优,因此,总执行时间更快,与 Min-Min 算法、基本蚁群算法相比分别减少了 16% ~29% 、10% ~17% 。

根据每个虚拟机处理任务的运行时间,利用公式 $LB=\sqrt{\frac{1}{m}\sum_{j=1}^{m}(c_j-\bar{c})^2}$ 计算其负载均衡度,3 种算法的实验结果如图 3 所示。

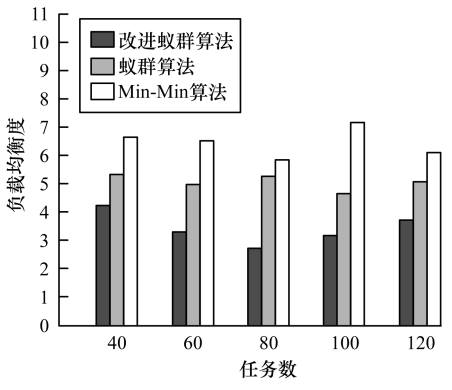

图 3　不同算法的负载均衡情况

由于 Min-Min 算法只把总执行时间作为优化目标,因此负载均衡度最差,本文设计的改进蚁群算法根据虚拟机的运行情况对局部信息素进行更新,优化了虚拟机处理任务时的负载均衡,对比基本蚁群算法,虚拟机工作的负载均衡效果更好。

4 结束语

通过蚁群算法处理云计算中的任务时,会遇到求解速度慢和负载不均衡的问题,本文通过添加权值的方法对信息素更新规则进行改进,利用每只蚂蚁所生成的解作为信息素增量的权重,加快算法在选择匹配方案时的求解效率,并采用动态更新挥发系数的方法,根据迭代次数自适应地对权值进行更新调整,提高算法的全局搜索能力,丰富解的多样性,改善了算法的收敛性,保证算法的综合性能。在对局部信息素更新时,加入衡量虚拟机负载的权重系数,使任务得到合理分配。实验结果表明,本文设计的改进算法能够较快地得到最优解,与 Min-Min 算法和基本蚁群算法相比,任务的总执行时间与负载问题得到了改善。为了更好地提高调度机制的综合性能,下一步将研究具有关联性的云任务调度与能耗优化问题。

参考文献

[1] ATIEWI S, YUSSOF S, EZANEE M. A comparative analysis of task scheduling algorithms of virtual machines in cloud environment[J]. Journal of Computer Science, 2015, 11(6): 804-812.

[2] WU K, LU P, ZHU Z. Distributed online scheduling and routing of multicast-oriented tasks for profit-driven cloud computing[J]. IEEE Communications Letters, 2016, 20(4): 684-687.

[3] BABU K R R, JOY A A, SAMUEL P. Load balancing of tasks in cloud computing environment based on bee colony algorithm [C]//Proceedings of the 5th International Conference on Advances in Computing and Communications. Washington D. C., USA: IEEE Press, 2016: 215-226.

[4] BIDOKI M Z, KARGAR M J. A social cloud computing: employing a bee colony algorithm for sharing and allocating tourism resources[J]. Modern Applied Science, 2016, 10(5): 177.

[5] ZUO Liyun, SHU Li DONG Shoubin. A multi-objective optimization scheduling method based on the Colony algorithm in cloud computing[J]. IEEE Access, 2015, 3: 2687-2699.

[6] GAO R, WU J. Dynamic load balancing strategy for cloud computing with ant colony optimization[J]. Future Internet, 2015, 7(4): 465-483.

[7] LAKSHMI R D, SRINIVASU N. A dynamic approach to task scheduling in cloud computing using genetic algorithm [J]. Journal of Theoretical & Applied Information Technology, 2016, 41(1).

[8] LIU C Y, ZOU C M, WU P. A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud Computing[C]//Proceedings of International Symposium on Distributed Computing and Applications to Business, Engineering and Science. Washington D. C., USA: IEEE Press, 2015: 68-72.

[9] XU Z. Task scheduling based on multi-objective genetic algorithm in cloud computing[J]. Journal of Information & Computational Science, 2015, 12(4): 1429-1438.

[10] 李智勇,陈少淼,杨 波,等. 异构云环境多目标 Memetic 优化任务调度方法[J]. 计算机学报, 2016, 39(2): 377-390.

[11] ABDULLAHI M, NGADI M A, ABDULHAMID S M. Symbiotic organism search optimization based task scheduling in cloud computing environment[J]. Future Generation Computer Systems, 2016, 56: 640-650.

[12] BALUSAMY B, KARTHIKEYAN K, SANGAIAH A K. Ant colony-based load balancing and fault recovery for cloud computing environment[J]. International Journal of Advanced Intelligence Paradigms, 2017, 9(2/3): 204.

[13] WU K, LU P, ZHU Z. Distributed online scheduling and routing of multicast-oriented tasks for profit-driven cloud computing[J]. IEEE Communications Letters, 2016, 20(4): 684-687.

[14] 陶晓玲,韦 毅,王 勇. 一种基于分层多代理的云计算负载均衡方法[J]. 电子学报, 2016, 44(9): 2106-2113.

[15] 李 琳,应 时,赵 翀,等. 基于蚁群算法的面向服务软件的部署优化方法[J]. 电子学报, 2016, 44(1): 123-129.

[16] YANG Q, CHEN W N, YU Z, et al. Adaptive multi-modal continuous ant colony optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(2): 191-205.

[17] 曾梦凡,陈思洋,张文茜,等. 利用蚁群算法生成覆盖表:探索与挖掘[J]. 软件学报, 2016, 27(4): 855-878.

编辑 索书志