

基于 TCBSA-ACO 算法在云计算任务分配中的研究

聂清彬^{1,2}, 霍敏霞¹, 曹耀钦¹

(1 重庆邮电大学 移通学院, 重庆 合川 401520; 2 四川大学 锦江学院, 四川 眉山 620860)

摘 要: 针对云计算中的任务分配问题, 提出一种基于建立时间成本负载约束函数的模拟退火蚁群算法(a restraint Function of Time Cost Load based on the Simulated Annealing ant colony Algorithm, TCBSA-ACO), 该算法结合云计算中任务分配的特点, 创新地通过建立时间成本约束函数和负载标准差函数分别改进信息素的更新和启发信息, 并用模拟退火算法对求出的解进行全局寻优. 利用 CloudSim 工具进行仿真测试, 与标准的蚁群算法 BACO 和最新的改进蚁群算法 DSFACO 做仿真对比, 实验结果表明 TCBSA-ACO 算法在云任务的执行时间, 成本, 系统负载均衡率方面均优于这两种算法, 提高了系统资源利用率.

关键词: 云计算; 蚁群算法; 负载均衡; 任务调度; 模拟退火

中图分类号: TP393

文献标识码: A

文章编号: 1000—7180(2016)06—0053—06

A Research of Time Cost Balance Simulated Annealing Ant Colony Algorithm for Task Allocation in Cloud Computing

NIE Qing-bin^{1,2}, HUO Min-xia¹, CAO Yao-qin¹

(1 College of Mobile Communication, Chongqing University of Posts and Telecom, Chongqing 401520, China;
2 Jinjiang College, Sichuan University, Meishan 620860, China)

Abstract: Aiming at solving the problem of task allocation in cloud computing, this paper proposes a restraint function of Time Cost Balance Simulated Annealing Ant Colony Algorithm (TCBSA-ACO). The TCBSA-ACO, combined with features of task allocation in cloud computing, improves information elements updating and inspiring factors creatively by establishing a time cost restraint function and a load standard deviation function, and achieves global optimization of all the answers by adopting the simulated annealing algorithm. Some simulation experiments are done by the tool CloudSim, and the results are compared with those of the standard ant colony algorithms BACO and the latest ant colony algorithm DSFACO. The comparison shows that this TCBSA-ACO algorithm is more efficient than the other two algorithms both in reducing time and cost of task execution and in keeping load balance, thus improves resource utilization in the system.

Key words: cloud computing; ant colony algorithm; load balance; task allocation; simulated annealing

1 引言

云计算成为近年来社会各界研究的热点, 它是由网格计算、并行计算、分布式技术发展而来的一种新型商业模式, 如何将任务合理地分配到资源节点上运行, 让整个云计算中心的资源节点得到充分利用, 让任务的执行的时间相对较短, 在保证较低成本

的同时兼顾负载均衡是我们追求的目标.

目前, 已有很多学者提出了不同的针对云计算中任务的调度算法, 文献[1]提出基于蚁群分组的蚁群算法, 该算法根据职能不同将蚂蚁进行分工, 根据分工不同的蚂蚁来完成对资源的局部搜寻以及全局的搜寻, 缩短了任务的完成时间; 文献[2]提出了在保持负载均衡的同时, 降低任务的完成时间, 把任务

一次性分配到虚拟机上作为被搜寻的目标,同时对任务执行时间和负载均衡度进行优化;文献[3]提出了基于最短任务延迟时间的改进蚁群算法(DSFA-CO),算法兼顾调度公平性和效率的前提下,最大化缩短任务延迟时间,提高任务执行效率.以上算法都不同程度地改进了云计算中资源调度的效率.本文在现有算法的基础之上加以改进,提出通过建立时间成本负载约束函数模型,并在蚁群算法后期引入模拟退火算法来优化路径选择的改进蚁群算法TCBSA-ACO^[4-5].

2 蚁群算法简介

2.1 蚁群算法原理

蚁群算法是模拟自然界的蚂蚁群体觅食行为的智能仿生算法.算法通过模仿蚁群在寻找食物过程中所表现出来的寻优能力被用来解决各种离散与复杂的组合优化问题^[6].

2.2 蚁群算法模型

将 n 只蚂蚁随机分配给 m 个城市,在城市 i 中的第 k 只蚂蚁在 t 时刻选择下一个城市 j 的概率为:

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{k \in \text{allowed}_k} [\tau_{ik}(t)]^\alpha [\eta_{ik}(t)]^\beta}, & j \in \text{allowed}_k \\ 0, & j \notin \text{allowed}_k \end{cases} \quad (1)$$

式中, $\tau_{ij}(t)$ 表示在 t 时刻路径 (i, j) 上的信息素浓度, $\eta_{ij}(t)$ 表示在 t 时刻路径 (i, j) 上的启发信息,定义 $\eta_{ij}(t) = \frac{1}{d_{ij}}$, d_{ij} 表示路径 (i, j) 的距离, β 为期望启发因子,表示启发函数重要程度,信息素启发因子为 α ,表示蚂蚁经过路径上留下信息素的影响程度, α 越大,表明信息素对后面蚂蚁路径选择影响越大,蚂蚁就会选择前面路径经过蚂蚁最多的路径,已经访问过的城市被加入禁忌表, allowed_k 表示第 k 只蚂蚁能够被允许经过的下一条路径的节点,是禁忌表之外蚂蚁还没走过的路径.设置蚂蚁的迭代次数为 nc ,最大迭代次数为 $nc_{\max}$.

3 模拟退火算法简介

模拟退火算法,是一种随机搜寻从而得到全局最优解的有效算法,其原理来自于现实世界中的固体退火的过程^[7-8].

在模拟退火算法中,某一个温度下,粒子的热运动处于平衡态的形式可以理解为:在开始时刻,粒子的初始点 X 的能量为 $E(X)$,随机产生一个随机数 ΔX , $X' = X + \Delta X$,计算出能量值为 $E(X')$,那么两

者的能量差 $\Delta E = E(X') - E(X)$,如果 $\Delta E < 0$,则将下一次退火的初始状态设置为 X' ;如果 $\Delta E > 0$,产生一个随机数 R , R 取值区间为 $[0, 1]$,则可以计算新状态被接受的概率 $P(\Delta E) = \exp(-\Delta E/K_B T)$,如果 $P(\Delta E) > R$,则接收新状态 X' ,否则放弃 X' ,用 X 作为下一次的初始状态.

4 云计算任务调度模型

4.1 云计算任务描述

云计算资源调度问题可以描述为:将 n 个独立的任务分配到 m 个虚拟机上执行 ($m < n$),其中 $t = \{t_1, t_2, \dots, t_n\}$ 表示任务的集合, $vm = \{vm_1, vm_2, \dots, vm_m\}$ 表示虚拟机的集合,每个子任务只能在一个虚拟机上运行,任务与虚拟机之间的分配关系可以用矩阵 d 来表示:

$$d = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{bmatrix} \quad (2)$$

式中, d_{ij} 表示子任务 t_i 在虚拟机 vm_j 的分配系, $i \in \{1, 2, \dots, n\}$, $j \in \{1, 2, \dots, m\}$, $d_{ij} \in \{0, 1\}$, $\sum_{j=1}^m d_{ij} = 1$ 如果子任务 t_i 在虚拟机 vm_j 上执行,则 $d_{ij} = 1$,否则 $d_{ij} = 0$.同理根据式(2)可以建立任务执行时间矩阵 et ,表示如下:

$$et = \begin{bmatrix} et_{11} & et_{12} & \cdots & et_{1n} \\ et_{21} & et_{22} & \cdots & et_{2n} \\ \vdots & \vdots & & \vdots \\ et_{m1} & et_{m2} & \cdots & et_{mn} \end{bmatrix} \quad (3)$$

et_{ij} 来表示子任务 t_i 在虚拟机 vm_j 上的执行时间,表示如下:

$$et_{ij} = \frac{\text{tasklength}_i}{ev_j} \quad (4)$$

式中, tasklength_i 表示任务 t_i 的长度, ev_j 表示虚拟机 vm_j 的执行能力,其计算公式如下:

$$ev_j = vm_num_j \times vm_mips_j + vm_mem_j + vm_bw_j \quad (5)$$

式中, vm_num_j 表示虚拟机 vm_j 的处理器数量; vm_mips_j 表示处理器 vm_j 的速度; vm_mem_j 表示虚拟机 vm_j 的内存; vm_bw_j 表示虚拟机 vm_j 的带宽.用 p 表示资源调度算法为用户提交的任务计算出来的资源分配方案,由于任务提交到虚拟机上都是并发执行的,所以整个云计算系统完成所有任务的时间跨度就是 et_{ij} 中的最大值,表示如下:

$$\text{finishtime}(p) = \max(et_{ij}) \quad (6)$$

4.2 时间约束函数模型

用 $\text{finishtime}_{\min}$ 表示用户提交的任务在具备最好性能的虚拟机上运行的时间,定义如下:

$$\text{finishtime}_{\min} = \frac{\sum_{i=1}^n \text{tasklength}_i}{m \times \max(ev_j)} \quad (7)$$

$\text{finishtime}_{\max}$ 表示用户提交的任务在具备最差性能的虚拟机上运行的时间,定义如下:

$$\text{finishtime}_{\max} = \frac{\sum_{i=1}^n \text{tasklength}_i}{m \times \min(ev_j)} \quad (8)$$

式中, $\max(ev_j)$ 表示最好性能虚拟机计算能力, $\min(ev_j)$ 表示最差性能虚拟机的计算能力. m 表示虚拟机的数量, 因此时间适应度函数 $f_{et}(p)$ 定义为:

$$f_{et}(p) = \frac{\text{finishtime}(p) - \text{finishtime}_{\min}}{\text{finishtime}_{\max} - \text{finishtime}_{\min}} \quad (9)$$

$f_{et}(p)$ 值越小, 表示总的任务执行时间越短.

4.3 成本约束函数模型

用 $\text{ecu}(\text{vm}_j)$ 表示虚拟机 vm_j 单位时间内运行任务所需消耗的成本, 结合上文每个虚拟机的运行时间 et_j , 就能计算出当前方案 p 完成所有任务所耗费的总成本 $\text{finisthcost}(p)$, 公式如下:

$$\text{finisthcost}(p) = \sum_{j=1}^m et_j \times \text{ecu}(\text{vm}_j) \quad (10)$$

用 $\text{finisthcost}_{\min}$ 表示用户提交的任务在 $\text{ecu}(\text{vm}_j)$ 最小的虚拟机上并发执行的费用;

$\text{finisthcost}_{\max}$ 表示用户提交的任务在 $\text{ecu}(\text{vm}_j)$ 最大的虚拟机上并发执行的费用.

$$\text{finisthcost}_{\min} = et_{\min} \times \min(\text{ecu}(\text{vm}_j)) \quad (11)$$

$$\text{finisthcost}_{\max} = et_{\max} \times \max(\text{ecu}(\text{vm}_j)) \quad (12)$$

因为成本适应度函数定义为:

$$f_{cost}(p) = \frac{\text{finisthcost}(p) - \text{finisthcost}_{\min}}{\text{finisthcost}_{\max} - \text{finisthcost}_{\min}} \quad (13)$$

因为 $f_{cost}(p)$ 的值越小, 任务完成所需的费用就越少. 因此通过式(9)和式(13)可以得到资源选择的约束函数为:

$$f(p) = t \times f_{et}(p) + c \times f_{cost}(p) \quad (14)$$

式中, $t \in [0, 1]$ 作为时间因子, $c \in [0, 1]$ 作为成本因子, $t + c = 1$, 当 $t = 0.5, c = 0.5$ 则该算法调度所产生的费用相对较低, 时间最短.

4.4 负载标准差模型

在云计算实际环境中, 影响负载的因素有虚拟

机 CPU 的使用率 cpu_ur , 内存的使用率 mem_ur , 网络带宽的利用率 bw_ur .

因此虚拟机 vm_j 的负载公式表示为:

$$\text{load}(\text{vm}_j) = a \times \text{cpu_ur} + b \times \text{mem_ur} + c \times \text{bw_ur} \quad (15)$$

其中 $a + b + c = 1$, a, b, c 分别表示 cpu 内存, 带宽的权值. 则整个云计算系统的平均负载为:

$$\text{load}_{\text{average}} = \frac{1}{m} \times \sum_{j=1}^m \text{load}(\text{vm}_j) \quad (16)$$

则整个云计算中心服务器负载均衡标准差 sdl 表示为:

$$sdl = \sqrt{\frac{\sum_{j=1}^m (\text{load}(\text{vm}_j) - \text{load}_{\text{average}})^2}{m}} \quad (17)$$

通过服务器的负载标准差的计算, 反应了系统的负载均衡状态, 标准差 sdl 越小, 反映出系统的大部分虚拟机负载值与系统平均负载值差值越小, 说明系统负载越均衡, 反之, 说明服务器负载越不均衡. 用负载均衡标准差 sdl 对式(1)中的 η 进行改进, 表示公式如下:

$$\eta = \frac{1}{sdl} \quad (18)$$

即负载均衡标准差越小, η 越大, 该虚拟机被选中的概率越大, 反之被选中的概率越小.

同理根据 sdl 可以计算出系统的负载不均衡度 LIB, 表示为:

$$\text{LIB} = \frac{sdl}{\sum_{j=1}^m \text{load}(\text{vm}_j)} \quad (19)$$

4.5 模拟退火算法中目标函数的建立

云计算环境下调度的目标就是最用户提交的任务实现资源的最优调度, 即调度的时间较短, 成本较低, 同时增强整个云计算系统负载均衡度, 将整个系统的负载不均衡度 LIB 和资源选择约束函数 $f(p)$ 作为调度算法的衡量指标, 目标函数 $G(Y)$ 定义如下:

$$G(Y) = \text{LIB} \times f(p) \quad (20)$$

5 算法流程

TCBSA-ACO 算法基础思想是在云环境中, 通过建立执行任务的时间以及所花费的成本约束函数模型和负载标准差函数模型, 用于改进蚁群算法信息素的更新和路径选择, 同时在蚁群算法后期加入模拟退火算法进一步优化已求出的解, 从而避免以往蚁群算法易陷入局部最优解的缺点, 进一步增强

蚁群算法的搜索性能,提高解的质量.具体算法流程如下所示.

(1) 对云计算中所有虚拟机进行初始化,设置信息素启发因子 α 和期望启发因子 β ,挥发因子 ρ ,任务数量 n ,虚拟机的个数 m ,迭代次数 nc 以及最大迭代次数 $nc_{\max}$,根据式(14)和式(19)定义云计算资源调度的时间成本约束模型 $f(p)$ 和负载不均衡度LIB,并根据式(20)定义目标函数 $G(Y)$.

(2) 将 n 个任务随机放置在 m 个虚拟机上进行搜索.

(3) 根据式(17)计算出负载差值 sdl 并作为启发信息,按状态转移概率式(1)来为下一个任务选择合适的虚拟机资源,改进公式如下:

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\frac{1}{sdl_{ij}}]^\beta}{\sum_{k \in \text{allowed}_k} [\tau_{ik}(t)]^\alpha [\frac{1}{sdl_{ik}}]^\beta}, & j \in \text{allowed}_k \\ 0, & j \notin \text{allowed}_k \end{cases} \quad (21)$$

(4) 并记下目前的最优解 Y ,并通过式(20)和式(19)计算出目标函数值 $G(Y)$ 以及当前前系统的负载不均衡度LIB.

(5) 利用模拟退火算法对上述求出的解做进一步优化以及信息素的更新,流程如下:

① 在蚁群算法的第 k 次迭代中计算出最优解 Y ,然后随机交换将两个虚拟机节点的任务,产生一个新解 Y' .

② 由模拟退火算法来进行判断是选择 Y 还是 Y' ,首先得到目标函数 $G(Y)$ 和 $G(Y')$,如果 $\Delta G = G(Y') - G(Y) < 0$,则用 Y' 代替 Y ,否则 $\exp(-\Delta G/T) > R$,也用 Y' 代替 Y ,否则就保留 Y 丢弃 Y' .退火温度按 $T(t+1) = \varphi T(t)$,其中 φ 为退火因子,退火温度为 T ,退火起始温度为 T_{start} ,最终的温度为 T_{finish} , R 表示产生的一个随机数, R 取值区间为 $[0,1]$.

③ 如果 $T < T_{\text{finish}}$ 算法继续迭代,若 $T \geq T_{\text{finish}}$ 迭代结束.

④ 对算法求出的最优解进行信息素更新,公式如下:

$$\Delta\tau_{ij}^k(t+1) = (1-\rho)\tau_{ij}^k(t) + \sum_{k=1}^m \Delta\tau_{ij}^k \quad (22)$$

其中, $\Delta\tau_{ij}^k$ 为每次迭代结束后最优解在路径 (i,j) 上释放的信息素, ρ 为信息素挥发因子.信息素增

量 $\Delta\tau_{ij}^k(t)$ 的定义是由来自式(14)的资源选择约束函数 $f(p)$ 决定的,信息素更新分为局部更新和全局更新.

① 局部信息素更新

当一只蚂蚁通过路径搜索,找到了相应的分配方案以后,路径上所有的虚拟机都将进行一次局部信息素更新.此时 $\Delta\tau_{ij}^k(t)$ 定义如下:

$$\Delta\tau_{ij}^k(t) = \frac{Q_1}{f(p_{nc})} \quad (23)$$

式中: Q_1 表示常量, $f(p_{nc})$ 表示第 k 只蚂蚁在第 nc 次迭代中为任务所找到的分配方案.

② 全局信息素更新

当所有蚂蚁完成路径搜索时,得到本次迭代中最优的分配方案,然后对全部虚拟资源节点进行全局信息素更新,此时 $\Delta\tau_{ij}(t)$ 定义如下:

$$\Delta\tau_{ij}^k(t) = \frac{Q_2}{\min(f(p_{nc}))} \quad (24)$$

式中 $\min(f(p_{nc}))$ 表示在第 nc 次迭代中为任务搜索到的最优分配方案, Q_2 为常量.

(6) 如迭代次数达到最大迭代次数,退出循环并返回算法结果,否则执行步骤(2).

6 实验仿真与结果分析

为验证本文算法对云任务资源调度的可行性与有效性,利用云计算仿真平台 CloudSim 进行仿真实验.

CloudSim 仿真步骤如下:

步骤一:初始化 CloudSim 包;

步骤二:创建数据中心;

步骤三:创建数据代理中心;

步骤四:创建虚拟机;

步骤五:创建云任务;

步骤六:调用任务调度策略,分配任务到虚拟机;

步骤七:启动仿真;

步骤八:仿真结束后分析结果.

在本仿真实验中,对 CloudSim 模拟器参数进行设置,设置虚拟机数量为 200 个,任务数量 50~350,任务长度为 5 000~15 000MI (Million Instruction),分配 25 个虚拟机,虚拟机性能参数 200~500 MIPS,内存为 512~2 048 MB,带宽为 2~4 M/s,虚拟机单位时间运行所消耗成本 $rcu(vm_j)$ 值 20\$, $T_0=10^5$, $T_{\text{end}}=10^4$.

对算法中的参数设置,

$$\alpha=1, \beta=1, \rho=0.05, t=0.5, c=0.5,$$

$$\Delta\tau_{ij}=0, nc_{\max}=30, \varphi=0.1$$

$$\text{虚拟机负载权值 } a=\frac{1}{3}, b=\frac{1}{3}, c=\frac{1}{3}.$$

在数据中心同时执行时间共享和时间共享策略,试验中在同等条件下将标准的蚁群算法 BACO,文献[6]中的改进蚁群算法 DSFACO 与本文 TCBSA-ACO 算法进行试验仿真,对比任务的执行时间、成本和系统的负载不均衡度,具体实验数据如下图所示,图中任务执行成本单位为美元(\$),执行时间单位为秒。

图1为BACO算法、DSFACO算法和TCBSA-ACO算法的任务调度时间图,三个算法在同等任务数量的情况下进行执行时间的对比,当任务数量在50到150左右时,执行的时间差别不大,这个主要是因为算法过程中引入模拟退火算法的随机性,导致刚开始算法的优越性不能马上表现出来,随着任务数量的增加,这样的随机性趋于均衡,BACO算法和DSFACO算法执行的时间大幅增加,而TCBSA-ACO执行的时间呈缓慢增加,执行时间明显更短,改进的算法体现出了优势,同时也说明算法中时间约束函数模型发挥了作用,表明算法具有很好的寻优能力。

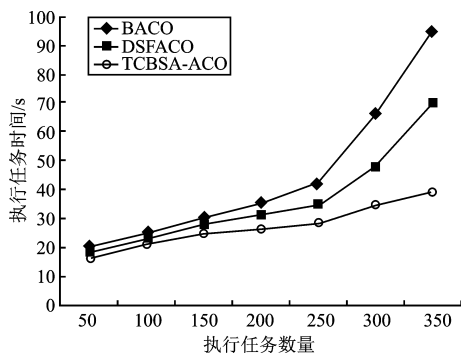


图1 任务执行时间对比图

图2为任务执行成本图,当任务在50到150之间时,三个算法执行相同数量任务的成本差别不大,随着任务数量的增加,B-ACO算法和DSFACO算法执行的成本增加明显,而TCBSA-ACO执行的成本增加幅度小,算法所产生的调度结果的任务执行费用少,说明算法中建立的成本适应度模型起了关键作用。

图3为BACO算法、DSFACO算法和TCBSA-ACO算法的系统负载不均衡度对比图,三个算法在同等任务数量的情况下进行负载不均衡度的对比,当任务数量较少时,三个算法的负载不均衡度相差

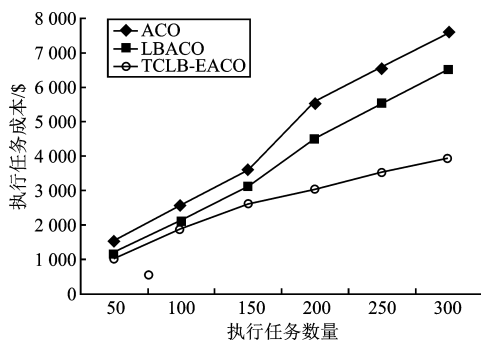


图2 任务执行成本对比图

不大,随着任务量的增加,BACO算法和DSFACO算法导致的系统负载不均衡度上升明显,而TCBSA-ACO算法不均衡度变化平稳,说明整个系统负载维持在一个相对均衡的状态。表明算法通过负载标准差模型改进启发信息,使算法在调度过程中将任务尽量往负载小的虚拟机上调度,系统负载更加均衡。结合图1中的任务执行时间对比结果分析,TCBSA-ACO算法执行任务时间短,负载均衡度高,这充分说明了虚拟机的利用率提高了,系统中虚拟机最大化的利用,让用户提交的任务的执行时间变小了,也印证了系统负载均衡度和任务的执行时间密切相关。

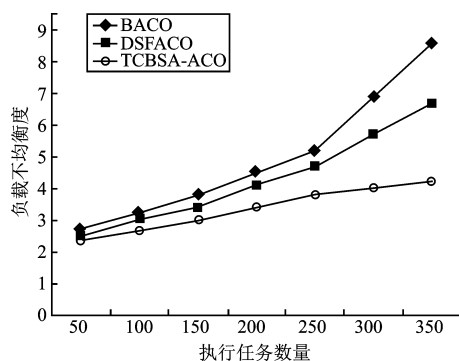


图3 系统负载不均衡度对比图

7 结束语

针对云计算资源调度的特点,通过改进调度算法,可以降低任务执行时间和成本,让整个云系统的资源节点负载更加均衡,本文通过对蚁群算法的改进,提出了TCBSA-ACO算法。

算法首先建立了基于时间,成本,负载的函数模型,对标准的蚁群算法中启发因子与信息素的更新进行改进,并且利用云仿真工具CloudSim进行试验仿真测试,与BACO算法以及最新提出的DSFACO算法在同等条件下对比执行任务所需的时间,

成本以及系统负载等情况

实验结果表明,TCBSA-ACO 算法性能明显优于 BACO 以及 DSFACO 算法,有效避免了系统资源负载过重,资源浪费的情况,降低任务执行的时间和成本,同时能让系统负载始终保持一个均衡的状态.

参考文献:

- [1] 张春艳,刘清林,孟珂. 基于蚁群算法的云计算任务分配[J]. 计算机应用,2012,32(5):1418-1420.
- [2] 查英华,杨静丽. 改进蚁群算法在云计算任务调度中的应用[J]. 计算机工程与设计,2013,34(5):1716-1719.
- [3] 张焕青,张学平. 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):31-40.
- [4] Kaur S, Verma A. An efficient approach to genetic algorithm for task scheduling in cloud computing environment[J]. Information Technology and Computer Science,2012,49(10):74-79.
- [5] Zheng Z N, Wang R. An approach for cloud resource scheduling based on parallel genetic algorithm [C]// Proc. of 3rd International Conference on Computer Re-

search and Development. China, Shanghai, 2011: 444-447.

- [6] Brotothi Mondal, Kousik Dasgupta, Paramartha Dutta. Load balancing in cloud computing using stochastic hill climbing-A soft computing approach[J]. Procedia Technology,2012,28(4):783-789.
- [7] Zhan S B. Huo H Y. Improved PSO-based task scheduling algorithm in cloud computing[J]. Information & Computational Science,2013,13(9):3821-3829.
- [8] Cristian Mateos, Elina Pacini, Carlos Garcia garion. An ACO-inspired algorithm for minimizing weighted flow time in cloud-based parameter sweep experiments [J]. Advances in Engineering Software,2013,56(2):38-50.

作者简介:

聂清彬 男,(1982-),硕士,讲师.研究方向为计算机软件与理论、云计算与物联网. E-mail: 270104318@qq.com.

霍敏霞 女,(1983-),硕士,讲师.研究方向为软件测试.

曹耀钦 男,(1962-),博士,教授,博士生导师.研究方向为计算机网络.

(上接第 52 页)

模拟和验证,仿真结果证明了模块功能的正确性,进而说明了新算法的可行性和性能优越性.

参考文献:

- [1] 任少杰,郝永生,许博浩. 射频识别技术综述[J]. 飞航导弹,2015,44(1):70-74.
- [2] 谢磊,殷亚凤,陈曦,等. RFID 数据管理:算法、协议与性能评测[J]. 计算机学报,2013,36(3):457-470.
- [3] 颜晓莲,陈庆奎,郝聚涛. 基于动态二进制的新防碰撞算法研究[J]. 小型微型计算机系统,2013,34(9):2148-2151.
- [4] 高金辉,郑晓彦. RFID 系统中二进制搜索防碰撞改进算法[J]. 计算机测量与控制,2013,39(2):2754-2756.
- [5] 李宝山,乔聪. 改进的二进制搜索防冲突算法[J]. 微电子学与计算机,2014,31(5):94-102.

[6] 魏峰. 远距离射频识别系统关键技术的研究[D]. 西安:西安电子科技大学. 2009.

[7] 周润景,苏良碧. 基于 Quartus II 的数字系统 Verilog HDL 设计实例详解 [M]. 北京:电子工业出版社,2010.

[8] 王蒙蒙,段本强. 基于按位排序的射频识别防碰撞算法[J]. 计算机工程与应用,2009,45(11):126-128.

作者简介:

任少杰 男,(1991-),硕士研究生.研究方向为信号采集与处理、导弹装备保障及运用. E-mail:RSJ5217@163.com.

郝永生 男,(1966-),副教授,硕士生导师.研究方向为信号采集与处理、导弹装备保障及运用.

许博浩 男,(1989-),硕士研究生.研究方向为信息采集与处理技术.