

基于改进蝙蝠算法的云计算任务调度研究

乔 良

(华北科技学院 计算机学院, 河北 三河 065201)

摘 要: 针对云计算虚拟机调度中存在的资源分配不均衡,蝙蝠算法收敛速度慢、寻优精度不高等缺点,提出了一种基于 K-means 和蝙蝠算法的云计算虚拟机调度算法. 利用 K-means 聚类对蝙蝠种群数据进行初始化,提高初始样本数据解的质量;利用 Powell 局部搜索算法对当前最优解进行局部搜索,提高收敛速度和精度;利用改进蝙蝠算法对虚拟机进行分配时,充分利用物理机上的资源,达到了最优化目标. 仿真实验表明,在虚拟机调度中,本文改进的调度算法与 K-means 均值调度算法相比,在物理节点的数量降低了 12% 左右,在系统资源综合利用率上平均提高了 11% 左右.

关键词: 蝙蝠算法; K-means; 虚拟机调度; Powell 局部搜索

中图分类号: TP391

文献标识码: A

文章编号: 1000—7180(2017)07—0027—06

Cloud Computing Task Scheduling Based on Improved Bat Algorithm

QIAO Liang

(School of Computer, North China Institute of Science & Technology, Sanhe 065201, China)

Abstract: For cloud computing resource allocation imbalance exists in the virtual machine scheduling, bat algorithm slow convergence speed and optimization accuracy is not high shortcomings, a method is proposed for calculating the virtual machine scheduling algorithm and K-means algorithm based on cloud bat. Algorithm using the K-means clustering initialized to bat population data, improve the quality of the initial solution of the sample data; By Powell local search algorithm for the optimal solution for the current local search and improve the convergence speed and accuracy; when using the improved bat algorithm to allocate the virtual machine, algorithm make full use of the resources on the physical machine to achieve the optimization goal. in the virtual machine scheduling, this paper improves the scheduling algorithm compared with the K-means mean scheduling algorithm, the number of the physical quantity node is reduced by about 12%, and the comprehensive utilization rate of system resources is increased by about 11%.

Key words: bat algorithm; K-means; virtual machine scheduling; powell local search

1 引言

云计算是一种新型的商业计算模式^[1],其关键的技术之一就是虚拟机调度,实现将虚拟机有效映射到相应的物理机上,达到物理机的负载均衡,有效利用现有资源. 针对虚拟机调度,国内外学者都进行了深入的研究. 文献[2]提出了一种经典的任务调度

算法,算法计算每个任务在各个资源上的完成时间,进行时间比较以此选择最短完成时间的任务优先调度;文献[3]将用户的优先级引入 Min-Min 算法来保证算法的负载均衡;文献[4]提出了一种基于多属性层次分析的虚拟机部署与调度策略,该算法将虚拟机按资源特点进行分类量化,根据量化后的权向量和服务器资源的使用记录,实现虚拟机的调度;文

献[5]在分组遗传算法的基础上,提出了基于多维资源协同聚合的虚拟机调度策略,对均衡资源分配具有一定的优势;文献[6]基于能耗比例模型的虚拟机调度算法,并采用“最近最小能耗比例优先”的策略进行调度,提高了云系统的能效.以上研究从不同的角度对虚拟机调度进行了优化和提升,但对云计算虚拟机调度中存在的资源分配不均衡.针对于此,本文在前人研究的基础上,提出一种基于 K-means 和蝙蝠算法的云计算虚拟机调度算法.

2 蝙蝠算法

蝙蝠算法(Bat Algorithm BA),是一种新型的群智能优化算法.在 BA 中将每个优化解得出即为搜索可行空间中的一个蝙蝠^[7-9].每个蝙蝠都有相应的适应度,BA 通过调整频率、脉冲发射率、响度在解空间中对最优蝙蝠进行搜索.在求解无约束优化问题上,BA 优于遗传算法和粒子群优化算法^[10],但也存在着容易陷入局部最优和收敛速度缓慢等问题,为了解决此问题,本文算法利用 K-means 聚类对蝙蝠种群数据进行初始化,提高初始样本解的质量,在得到最优蝙蝠后对其进行 Powell 局部搜索,加快算法收敛的速度.基本蝙蝠算法的步骤如下:

步骤 1 参数初始化:设最大迭代次数为 $\max$,搜索精度 ϵ ,脉冲频率的最大最小值分别为 $f_{\max}$ 、 $f_{\min}$,蝙蝠的初始化位置为 $X_i (i = 1, 2, 3, \dots, N)$,搜寻当前蝙蝠群中最优蝙蝠位置为 X^* ;

步骤 2 在种群迭代中根据以下公式更新蝙蝠的速度和位置:

$$f_i = f_{\min} + (f_{\max} - f_{\min})\alpha \quad (1)$$

$$v_i^{t+1} = v_i^t + (X_i^t - X^*)f_i \quad (2)$$

$$X_i^{t+1} = X_i^t + v_i^{t+1} \quad (3)$$

上式中, f_i 为第 i 只蝙蝠的脉冲频率, α 为 $[0, 1]$ 上均分分布的随机数, v_i^t 、 X_i^t 分别为第 i 只蝙蝠 t 时刻的速度和位置;

步骤 3 生成随机数 R_1 ,如果 $R_1 > r_i$,则对当前最优蝙蝠位置按照以下式(4)进行更新, r_i 为第 i 只蝙蝠的脉冲频度;

$$X_{\text{new}} = X_{\text{old}} + \delta A^t \quad (4)$$

步骤 4 生成随机数 R_2 ,如果 $R_2 < A_i$ 且 $f(X_{\text{new}}) < f(X^*)$,当前蝙蝠位置可以更新的话,将按照公式(5)和(6)则对当前最优蝙蝠位置进行更新, A_i 为第 i 只蝙蝠的脉冲强度;

$$f_i^{t+1} = f_{\max}[1 - \exp(-\sigma \cdot t)] \quad (5)$$

$$A_i^{t+1} = \omega A_i^t \quad (6)$$

f_i^{t+1} 为第 i 只蝙蝠 $t+1$ 时刻的脉冲频度, σ 为脉冲频度的增加系数, A_i^{t+1} 为第 i 只蝙蝠 $t+1$ 时刻的脉冲音强, ω 为脉冲音强的衰减系数;

步骤 5 对蝙蝠群体按照适应度值进行评估,记录最优位置和适应度值;

步骤 6 迭代寻优,转到步骤 2,直到满足搜索精度或达到最大迭代次数,转到步骤 7;

步骤 7 终止算法,输出最优函数值和蝙蝠的最优位置.

3 改进蝙蝠算法

3.1 基于 K-means 聚类改进蝙蝠算法

原始蝙蝠算法的初始化是用随机函数产生的,随机函数不能保证蝙蝠种群原始数据分布均匀,而寻优精度受种群分布的影响,为了提高初始样本解的质量,本文利用 K-means 聚类对蝙蝠种群数据进行初始化,改进步骤如下:

步骤 1 随机产生 n 个蝙蝠个体作为待聚类个体的初始聚类中心,设定聚类中心的变化精度阈值 φ ;

步骤 2 随机产生 m 个蝙蝠个体,计算每个蝙蝠个体到初始聚类中心的欧式距离,按距离的大小将每个蝙蝠个体安排到距离最近的聚类中去,通过距离大小衡量聚类结果的优劣,聚类的距离越近,说明其相似度就越大,聚类效果越好;

步骤 3 重新计算 n 个聚类簇的中心,替代之前的聚类簇中心,重复执行步骤 2 和 3,直到相邻聚类中心变化范围小于设定的阈值;

步骤 4 输出 n 个中心点,生成 n 个蝙蝠初始种群.

3.2 基于 Powell 局部搜索改进蝙蝠算法

Powell 算法又称方向加速算法,是利用共轭方向可以加快收敛速度的性质形成的一种搜索算法.该方法不需要对目标函数进行求导,当目标函数的导数不连续时也能应用,并且不需要计算梯度,搜索精度也较高,因此 Powell 算法是一种十分有效的直接搜索法,但其缺点是对初始点要求颇高,本文利用 K-means 聚类对蝙蝠种群数据进行初始化,解决 Powell 算法对初始点的要求. Powell 算法基本步骤如下:

步骤 1 根据 K-means 聚类对蝙蝠种群的初始化结果,设定初始点为 $c^{(0)}$,搜索精度为 δ ,给定 n 个初始蝙蝠种群的无关搜索方向 $d^{(i)} (i = 0, 1, 2, \dots, n-1)$,一般取 n 个坐标轴方向, $j = 0$;

步骤2 令 $c^{(0)} = c^{(j)}$, 从 $c^{(0)}$ 开始依次沿 $d^{(i)}$ ($i = 0, 1, 2, \dots, n-1$) 这 n 个搜索方向进行一维搜索, 可得 $c^{(i)}$ ($i = 1, 2, \dots, n$):

$$f(c^{(i)} + \omega_i d^{(i)}) = \min f(c^{(i)} + \omega d^{(i)}) \quad (7)$$

$$c^{(i+1)} = c^{(i)} + \omega_i d^{(i)}, (i = 0, 1, 2, \dots, n) \quad (8)$$

上式中 ω_i 为一维最优化问题的解, 由线性搜索得到, ω 为步长;

步骤3 设 $d^{(n)} = c^{(n)} - c^{(0)}$, 假如 $\|d^{(n)}\| \leq \delta$, 求得解 $c^{(n)}$ 后结束计算; 否则, 从 $c^{(n)}$ 出发沿 $d^{(n)}$ 方向进行线性搜索直到得到 $c^{(n+1)}$;

步骤4 确定搜索方向, 按照以下公式(9) 计算指标 m :

$$f(c^{(m)}) - f(c^{(m+1)}) = \max_{0 \leq i \leq n-1} \{f(c^{(i)}) - f(c^{(i+1)})\} \quad (9)$$

步骤5 判断 $f(c^{(0)}) - 2f(c^{(n)}) + f(2c^{(n)} - c^{(0)}) < 2[f(c^{(m)}) - f(c^{(m+1)})]$ 是否成立, 若成立执行步骤6, 否则执行步骤7;

步骤6 说明 $d^{(0)}, d^{(1)}, \dots, d^{(n-1)}$ 线性相关, 需要调整搜索方向, 从点 $c^{(n)}$ 出发沿 $d^{(n)}$ 进行一维搜索, 求得 ω_n , 使 $f(c^{(n)} + \omega_n d^{(n)}) = \min f(c^{(n)} + \omega d^{(n)})$, 令 $c^{(j+1)} = c^{(n)} + \omega_n d^{(n)}$, $d^{(j)} = d^{(j+1)}$ ($j = m, m+1, \dots, n-1$), $j = j+1$, 返回步骤2;

步骤7 说明 $d^{(0)}, d^{(1)}, \dots, d^{(n-1)}$ 线性无关, 不调整搜索方向, 令 $c^{(j+1)} = c^{(n)}$, $j = j+1$, 返回步骤2.

4 云计算虚拟调度模型

云计算数据中心有 β 个物理节点, 有 α 个虚拟机发送调度请求, 每个物理节点的资源向量分为已使用的资源向量和空闲的资源向量, 物理节点上的资源是多维的, 包括 CPU、内存、带宽和 I/O 等, 每个物理节点已使用资源等于分配到该节点的虚拟机资源之和^[11-12]. 设虚拟机集合为 $V = \{V_i \mid 1 \leq i \leq \alpha\}$, 物理节点集合为 $P = \{P_j \mid 1 \leq j \leq \beta\}$.

4.1 调度思想

本文提出的虚拟机调度方法的基本思想是将每个虚拟机分配到距离最近的物理节点上, 通过多次迭代, 每个虚拟机都与所在的物理节点最近. 用资源相关系数衡量虚拟机与物理节点的距离, 相关系数越小, 说明虚拟机所需资源与物理节点拥有的资源之间具有更多的互补, 在空闲资源满足条件的情况下, 将其分配到相关系数小的物理节点. 采用皮尔逊相关系数计算虚拟资源与物理节点之间的相关性,

取值范围为 $[-1, 1]$:

$$\bar{\omega}_{vp} = \frac{\sum_{k=1}^t (V_k - V)(\bar{P}_k - \bar{P})}{\sqrt{\sum_{k=1}^t (P_k - \bar{P})^2} \sqrt{\sum_{k=1}^t (V_k - \bar{V})^2}} \quad (10)$$

将虚拟机与物理节点的距离定义为:

$$d_{vp} = \frac{1 + \bar{\omega}_{vp}}{2} \quad (11)$$

则虚拟机与物理节点的距离取值范围为 $[0, 1]$. 本文将虚拟机到不活跃物理节点的距离定义为这最大, 将其取值为 1, 表示只有当虚拟机无法放置时才分配到新物理节点.

4.2 蝙蝠编码规则

蝙蝠算法采用实数据编码, 一个编码对应一个可行解. 本文采用 β 个物理节点为初始聚类中心, 这样聚类对应的物理节点限制了聚类的大小, 简化了对资源足的考虑, 确定的聚类减少了迁移的次数, 使资源的分配更加稳定.

每个蝙蝠由 β 个聚类中心构成. 设当前虚拟机分为 β 类, 每个数据为 d 维特征, 用于实数进行编码, 以 k 均值聚类中心作为寻优变量, 每个可行解是由 β 个聚类中心构成, 由于解的维数是 d 维, 这里可行解的位置是 $\beta \times d$ 维向量, 可行解的编码示例, 其结构如表 1 所示.

表 1 蝙蝠算法的编码结构

Z_{11}		Z_{1d}		Z_{k1}	Z_{k2}		Z_{kd}
----------	-------	----------	-------	----------	----------	-------	----------

表中 $Z_{\beta 1}, Z_{\beta 2}, \dots, Z_{\beta d}$ 代表第 β 类的 d 维聚类中心.

4.3 基于改进蝙蝠算法的虚拟机调度步骤

(1) 初始化改进蝙蝠算法的基本参数; 给定含有 n 个虚拟机对象的数据集 T 和物理节点数 m , 利用 K-means 聚类对蝙蝠种群数据进行初始化, 将聚类中心作为蝙蝠的位置编码, 反复进行多次, 生成 N 个初始蝙蝠, 令 $s = 1$;

(2) 初始化蝙蝠种群的速度、脉冲频率、脉冲响度和脉冲发射速率;

(3) 根据适应度公式计算每个蝙蝠的适应度值, 保留最优解, 并利用速度和位置公式对其他蝙蝠进行更新;

(4) 产生一个随机数 R_1 , if ($R_1 > r_i$) 则对当前群体中最佳蝙蝠位置进行随机扰动得到替换当前蝙蝠的位置;

(5) 产生随机数 R_2 , if ($R_2 < A_i$ and $f(X_{\text{new}}) < f(X^*)$), 则接受这个新解, 利用音强 A_i 和脉冲发生

率 r_i 公式减少音强,增大脉冲发生率;否则,不更新音强 A_i 和脉冲发生率 r_i ,转到步骤 6;

(6) 评估蝙蝠群体,对当前群体中的最优位置按照 3.2 节的改进进行 Powell 局部搜索;

(7) 判断算法是否达到结束条件(有满足条件的位置或达到最大迭代次数),若满足,则转入步骤 8;否则,令 $s = s + 1$,返回步骤 3;

(8) 输出最优的蝙蝠位置对应的数据聚类结果,算法结束.

5 仿真实验

为了验证本文改进算法的有效性,设计了两个实验方案:一是选取测试函数对比其他算法,验证本算法寻优的优越性;二是采用 Cloudsim3.0 云平台软件进行对本文算法的实际调度性能进行模拟仿真,验证本算法在云计算虚拟调度中的性能,仿真是在 windows 7 系统下,CPU:i3-3240@3.4 GHz, RAM:4 GB.

5.1 寻优测试

从文献[13]选取四个标准函数如表 2 所示,利

表 3 三种算法对四个标准函数的寻优结果

函数	算法	最优值	最差值	平均值	标准差
Sphere	BA	4.32e-6	9.45e-6	5.285e-6	7.12e-7
	PSO	1.48e-2	4.41e-1	1.72e-1	1.9e-1
	本文算法	2.18e-11	3.29e-10	1.23e-10	1.07e-10
Rosenbrock	BA	12.537 12	39.179 25	21.615 76	6.905 25
	PSO	19.759 24	133.340 19	75.831 47	33.124 21
	本文算法	7.48e-5	1.39	1.44e-1	4.08e-1
Rastrigin	BA	5.001 032	19.900 087	10.958 155	5.118 971
	PSO	19.270 017	61.800 127	41.070 032	14.636 472
	本文算法	1.49e-6	2.17e-3	7.01e-4	1.05e-5
Ackley	BA	2.320 056	3.036 877	3.799 826	4.54e-1
	PSO	5.360 042	15.802 238	20.000 186	4.964 796
	本文算法	1.29e-7	3.38e-2	7.495e-3	1.24e-2

从表 3 可以看出,与标准蝙蝠算法(BA)和粒子优化算法(PSO)相比,本文算法对 Sphere 等四个标准函数都有较好的最优值、最差值、平均值和标准差,粒子优化算法(PSO)对四个标准函数的寻优误差较大,而标准蝙蝠算法(BA)只对简单的测试函数 Sphere 有较高的寻优结果.为了全面的反映本文算法的性能,以下给出了三个算法对四个标准函数的寻优收敛曲线,如图 1 中的(a)-(d)所示:

由图 1 可以看出,标准蝙蝠算法(BA)和粒子优

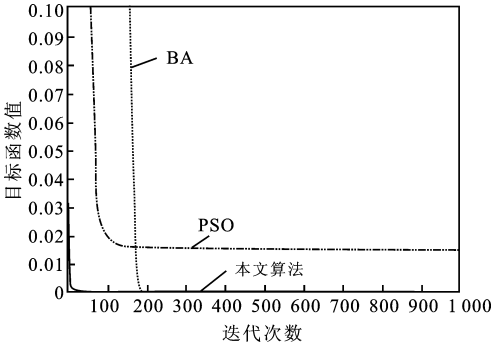
用本文算法、粒子优化算法(PSO)和标准蝙蝠算法(BA)等三个算法对四个标准函数进行求解来测试算法的寻优性能.参数设置如下:标准蝙蝠算法(BA)的最大脉冲 $f^0 = 0.8$,脉冲频率范围为 $[-1, 1]$,最大脉冲音强 $A = 0.25$,脉冲频度的增加系数 $\sigma = 0.02$,脉冲音强的衰减系数 $\omega = 0.9$,本文算法的参数与标准蝙蝠算法一致.根据文献[14]所提出的线性降低惯性权重, $c_1 = c_2 = 1.496 2$, $\omega_{\max} = 0.9$, $\omega_{\min} = 0.4$,设种群规模 $n = 50$,最大迭代次数 $\max = 1\,000$.根据以上参数设置,每种算法运行 25 次,表 3 为三种算法对四个标准函数的寻优结果表明.

表 2 四个标准函数

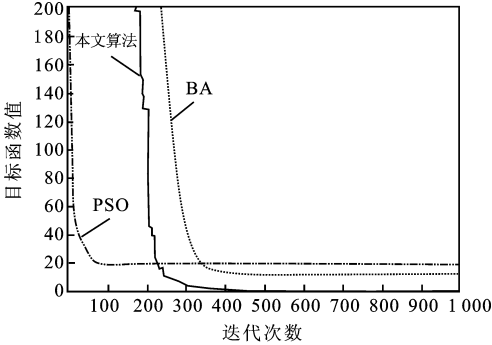
函数名	维数	搜索范围	最优值
Sphere	20	$[-100, 100]$	0
Rosenbrock	20	$[-30, 30]$	0
Rastrigin	20	$[-5.12, 5.12]$	0
Ackley	20	$[-30, 30]$	0

化算法(PSO)的对前三种标准函数的总体寻优性能相似,对高维复杂空间的寻优效果不是很好,在对多峰函数 Ackley 进行寻优测试时,随着迭代次数的增加,粒子优化算法(PSO)的目标函数值下降速度过缓,寻优精度不佳,收敛速度过慢;对于多峰 Rastrigin、Ackley 函数,标准蝙蝠算法(BA)表现出收敛过快且易早熟,而本文改进的蝙蝠算法不管对于单峰函数 Sphere、Rosenbrock 还是多峰函数 Rastrigin、Ackley 函数,在整个迭代次数范围内表现出了良好

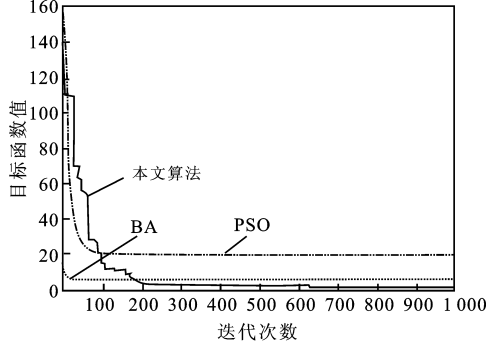
的性能,目标函数值曲线下落速度快,寻优精度高,随着迭代次数的增加,寻优都能得到理论最优值.



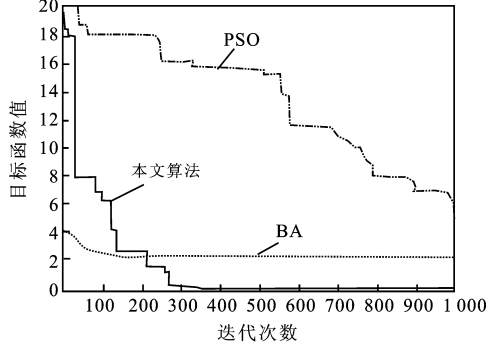
(a)三种算法对Sphere函数的寻优曲线



(b)三种算法对Rosenbrock函数的寻优曲线



(c)三种算法对Rastrigin函数的寻优曲线



(d)三种算法对Ackley函数的寻优曲线

图 1 三个算法对四个标准函数的寻优收敛曲线

5.2 云计算调度测试

CloudSim 是澳大利亚墨尔本大学的网络实验室和 Gridbus 项目宣布推出的云计算仿真软件. 本

仿真实验将虚拟机分别设置为 100,200,300,400,500,每个实例取 10 次运算的平均值,所需物理节点数量和系统资源综合利用率两方面与 K-means 均值调度算法进行比较.

(1)物理节点数据对比

从图 2 可以看出,本文调度方法的物理节点的数量比 K-means 均值调度方法平均降低 12%左右,有效的减少了资金投入,物理机节点的能耗也降低,使数据中心具有高密度、低成本的优点. 主要原因是本文方法中通过聚类方法将虚拟机分配到服务器中,并且将虚拟机到不活跃物理节点的距离设置为这最大,表示当只有当虚拟机无法放置时才分配到新物理节点,这样有效的提高了数据中心物理节点的分配密度,降低了物理节点的节点数量.

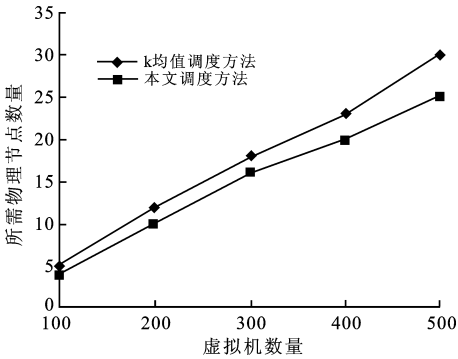


图 2 物理节点使用数量对比

(2)系统资源综合利用率比较

图 3 显示本文调度方法系统资源综合利用率比 K 均值调度方法平均提高了 11%左右,有效的提高了资源的综合利用率. 其原因主要是因为本文调度方法在调度时考虑了虚拟机与服务器资源的互补性,有效的防止出一种资源过于饱和导致其他资源不能使用的情况,通过资源互补分配虚拟机有效的提高了资源的综合利用率.

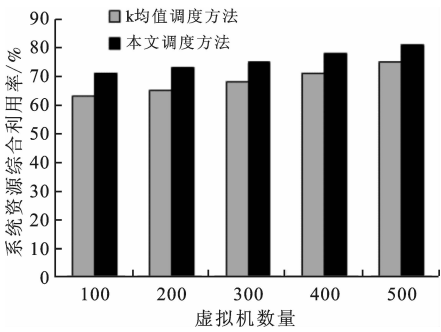


图 3 系统资源综合利用率对比

6 结束语

本文提出了基于 K-means 均值和蝙蝠混合优化的云计算虚拟调度算法,利用 K-means 聚类对蝙蝠种群数据进行初始化,提高初始样本数据解的质量,利用 Powell 局部搜索算法对当前最优解进行局部搜索,提高收敛速度和精度,利用改进蝙蝠算法对虚拟机进行分配时,与 K-means 均值调度算法相比,不仅降低了物量节点的数量还提高了系统资源综合利用率,达到了最优化目标.

参考文献:

[1] Lin W, Qi D. Research on resource self-organizing model for cloud computing[C]// Internet Technology and Applications, 2010 International Conference on, Wuhan, China, IEEE, 2010; 1-5.

[2] Bawa R K, Sharma G. Modified min-min heuristic for job scheduling based on QoS in Grid environment[C]// IEEE International Conference on Information Management in the Knowledge Economy, India, Chandigarh, 2013; 166-171.

[3] Chen H K, Wang F, He L N, et al. User-priority min-min scheduling algorithm for load balancing in cloud computing [C]// 2013 National Conference on Parallel Computing Technologies. India, Bangalore, 2013; 1-8.

[4] 庄威, 桂小林, 林建材, 等. 云环境下基于多属性层次分

析的虚拟机部署与调度策略[J]. 西安交通大学学报, 2013, 47(2): 28-32.

[5] 袁爱平, 万灿军. 云环境下基于改进遗传算法的虚拟机调度策略[J]. 计算机应用, 2014, 34(2): 357-359.

[6] 肖鹏, 刘洞波, 屈喜龙. 云计算中基于能耗比例模型的虚拟机调度算法[J]. 电子学报, 2015, 43(2): 305-310.

[7] 李煜, 马良. 新型全局优化蝙蝠算法[J]. 计算机科学, 2013, 40(9): 225-228.

[8] 金伟健, 王春枝. 基于蝙蝠算法的云计算资源分配研究[J]. 计算机应用研究, 2015, 32(4): 1184-1187.

[9] 全雪皎, 孟凡荣, 王志晓. 对 k-mean 初始聚类中心的优化[J]. 计算机工程与设计, 2011, 31(8): 165-167.

[10] Yang X S. Nature inspired meta-heuristic algorithms [M]. 2nd ed. UK: Luniver Press, 2010; 97-104.

[11] 李乔, 郑啸. 云计算研究现状综述[J]. 计算机科学, 2011, 38(4): 32-37.

[12] 徐文忠, 彭志平, 左敬龙. 基于遗传算法的云计算资源调度策略研究[J]. 计算机测量与控制, 2015, 23(5): 1653-1656.

[13] 李煜, 马良. 新型全局优化蝙蝠算法[J]. 计算机科学, 2013, 40(9): 225-227.

[14] 冯辉宗, 彭丹, 袁荣棣. 基于 PSO-SVM 的发动机故障诊断研究[J]. 计算机测量与控制, 2014, 22(2): 355-357.

作者简介:

乔良男, (1980-), 博士, 讲师. 研究方向为数据挖掘、云集计算任务调度、物联网. E-mail: qiaoliang80@sina.com.

(上接第 26 页)

[5] Ye X, Ding D, Yu L. A hardware-oriented ime algorithm and its implementation for hevc[C]// Visual Communications and Image Processing Conference. [s. l], 2014.

[6] Song L, Tang X, Zhang W, et al. The sjtu 4k video sequence dataset[C]// 2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX). [s. l], 2013; 34-35.

[7] Bjontegaard G. Calculation of average PSNR differ-

ences between RD-curves[D]. Austin, Texas, USA, 2001, VCEG M33.

作者简介:

胡沥男, (1991-), 硕士研究生. 研究方向为数字电路设计与实现. E-mail: 188001970103@163.com.

何卫锋男, (1976-), 副研究员. 研究方向为 SoC 设计与系统集成.