基于动态优先级和萤火虫行为的云任务调度算法*

刘亚秋[1,2]，赵青华[1,2]，景维鹏[1,2†]

(1. 东北林业大学 信息与计算机工程学院，哈尔滨 150040; 2. 黑龙江省林业生态大数据存储与高性能(云)计算工程技术研究中心，哈尔滨 150040)

摘 要: 针对云环境下优先级任务调度容易出现的负载失衡问题，提出一种基于动态优先级和萤火虫行为的云任务调度算法(TS-PFB)。该算法根据任务价值密度与执行紧迫性计算出动态优先级; 模拟萤火虫行为，结合吸引度(ECT)和荧光亮度(负载约束)给出决策变量ρ; 再按优先级高低依次将任务调度到最大ρ值所对应的可行VM上。实验结果表明，相比于Min-Min、Max-Min、HBB-LB算法，该算法减少了总任务的完成时间，均衡了VM的负载，降低了任务截止期错失率。

关键词: 动态优先级; 萤火虫行为; 负载均衡; 任务调度; 云计算

中图分类号: TP301.6 **文献标志码:** A **文章编号:** 1001-3695(2015)04-1040-04

doi: 10.3969/j.issn.1001-3695.2015.04.019

Task scheduling algorithm based on dynamic priority and firefly behavior in cloud computing

LIU Ya-qiu[1,2], ZHAO Qing-hua[1,2], JING Wei-peng[1,2†]

(1. *College of Information & Computer Engineering, Northeast Forestry University, Harbin* 150040, *China*; 2. *Heilongjiang Province Engineering Technology Research Center for Forestry Ecological Big Data Storage & High Performance Computing*(*Cloud Computing*), *Harbin* 150040, *China*)

Abstract: Considering the load balancing scheduling problem of the priority tasks in cloud environment, this paper proposed a task scheduling algorithm based on dynamic priority and firefly behavior(TS-PFB), which calculated the priority by the value of density and the urgency of execution, and gave the decision variables ρ combined attractiveness(ECT) and fluorescence intensity(load restraint) by simulating the firefly behavior, and finally scheduled tasks in turn onto the available VM with maximum value of ρaccording to its priority. Experimental results show that, compared with the Min-Min and Max-Min and HBB-LB, the TS-PFB reduces the makespan, balances the load of VMs, reduces the miss ratio of tasks in deadline.

Key words: dynamic priority; firefly behavior; load balancing; task scheduling; cloud computing

1 概述

云计算是在网格计算、分布式计算、并行计算和虚拟化等技术基础上发展起来的一种基于互联网的计算模式，它将大量通过网络连接的计算资源、存储资源与软件资源整合起来，构成共享虚拟资源池，为用户提供廉价的按需服务，包括基础设施即服务(IaaS)、平台即服务(PaaS)和软件即服务(SaaS)[1,2]。

优先级任务调度是云计算任务调度研究的热点之一。文献[3]利用层次分析法对优先级任务进行了调度。文献[4]在Hadoop平台上实现了优先级任务的调度。传统的Min-Min与Max-Min算法给予最小或最大任务更高的优先级。虽然以上算法都减少了总任务的完成时间，但都没有考虑价值与截止时间对优先级的影响，Min-Min算法还容易出现负载失衡。为了避免大任务长时间得不到执行，文献[5]在Min-Min算法中引进了动态优先级，任务优先级随着等待时间的增加而增大，而优先级越大，任务能够越快得到执行。文献[6]为了避免优先级无限大情况的出现，按任务优先级分组，组内优先级相等，然后优先调度截止时间小的任务。以上两种任务调度算法虽然都提高了资源的利用率，但都没有考虑任务价值对优先级的影响。文献[7]综合任务的剩余价值密度和执行紧迫性，提出了一种单机环境下动态优先级实时任务调度算法，但该算法不适用于分布式的云环境。

高效的任务调度算法在减小总任务调度时间的同时也应能够均衡VM的负载。文献[8]利用VM迁移技术实现了物理机的负载均衡。文献[9]运用指数平滑法预测方法实现了VM资源的动态负载均衡。文献[10]在VM记录的历史数据和当前状态的基础上运用遗传算法实现了VM的动态负载均衡。

收稿日期: 2014-03-21; 修回日期: 2014-05-05 基金项目: 高等学校博士学科点专项科研基金资助项目(20120062110012); 国家自然科学基金资助项目(31370565)

作者简介: 刘亚秋(1971-)，男，辽宁法库人，教授，博导，主要研究方向为高性能计算、嵌入式计算; 赵青华(1990-)，男，硕士研究生，主要研究方向为云计算、调度算法; 景维鹏(1979-)，男(通信作者)，副教授，博士，主要研究方向分布式计算、容错计算(nefujwp@163.com).

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

文献［11］模仿蜜蜂采蜜行为提出了一种动态负载均衡算法 HBB-LB，该算法在负载失衡后，以过载 VM 的任务为蜜蜂，以轻载 VM 为蜜源进行信息采集，然后依据这些信息实现任务迁移以达到负载均衡。以上动态负载均衡策略都能够均衡负载，适用于复杂任务调度，但实现较为复杂，会增大开销。静态负载均衡方面，文献［12］利用蚁群算法实现了负载均衡调度。文献［13］在 Max-Min 算法的基础上增加了负载约束。文献［14］在预调度 Min-Min 算法的基础上，选择过载 VM 上最小任务，再调度到能使总任务完成时间缩短的 VM 上。以上静态调度算法实现较为简单，能够较好地实现负载均衡，适用于简单任务调度。

综合考虑任务的价值与截止时间对优先级的影响，同时平衡 VM 负载，本文提出了一种基于优先级和萤火虫行为的任务调度算法（task scheduling algorithm based on dynamic priority and firefly behavior in cloud computing，TS-PFB）。该算法依据任务的价值密度与执行紧迫性构建了随着时间增长的动态优先级，然后按优先级高低依次调度任务，通过模拟萤火虫行为，以虚拟机和待调度的任务为萤火虫，计算出综合考虑吸引度（最早完成时间）和荧光亮度（负载约束）的决策变量 ρ，最后将任务调度到最大 ρ 值所对应的 VM 上。

2　相关模型

当前的云计算环境大部分采用谷歌提出的 MapReduce 编程模式，在开源的 Hadoop 平台上，通过 Map（映射）和 Reduce（化简）两阶段，将一个或几个较大的任务分割成为若干较小的子任务，当子任务队列达到一定的阈值后再统一进行任务调度，将任务按照约束条件分配到合适的虚拟机上执行，最后返回运行结果。

2.1　任务调度模型

本文只考虑 n 个不相关独立计算子任务的调度情况，这些任务将分配给 m 个虚拟资源节点执行（$m<n$）。

n 个子任务的集合表示为 $T(n)=\{t_1,t_2,\cdots,t_n\}$，其中 $n\in N$，用 t_i 表示集合中第 i 个子任务（$i=1,2,\cdots,n$）。而任务 t_i 可以由其对应的属性向量 $t_i=(t^i_{\mathrm{id}},t^i_{\mathrm{mi}},t^i_{\mathrm{fee}},t^i_{\mathrm{deadline}},t^i_{\mathrm{memory}},t^i_{\mathrm{bw}},t^i_{\mathrm{submit}})$ 来表示，其中，t^i_{id} 为任务的唯一的标志号；t^i_{mi} 表示任务大小，代表任务的百万条指令数（MI）；t^i_{fee} 是用户期望任务的费用；t^i_{deadline} 是用户期望任务的截止时间；t^i_{memory} 表示任务对内存大小的需求；t^i_{bw} 是任务对带宽的需求；t^i_{submit} 为用户提交任务的时间。

m 个虚拟机资源的集合表示为 $VM(m)=\{vm_1,vm_2,\cdots,vm_m\}$（$m\in N$），用 vm_j 表示第 j 个虚拟机（$j=1,2,\cdots,m$），而虚拟机资源可由其对应的属性向量 $vm_j=(vm^j_{\mathrm{id}},vm^j_{\mathrm{capacity}},vm^j_{\mathrm{bw}},vm^j_{\mathrm{memory}})$ 来表示。其中 vm^j_{id} 是虚拟机所在数据中心的唯一标志号；vm^j_{capacity} 是虚拟机资源的处理能力，用每秒百万条指令数 MIPS 表示；vm^j_{bw} 是虚拟机能提供的带宽；vm^j_{memory} 代表虚拟机内存大小。

用 ETC_{ij} 表示子任务 t_i 在虚拟资源 vm_j 上的期望执行时间，则其可表示为

$$\mathrm{ETC}_{ij}=\frac{t^i_{\mathrm{mi}}}{vm^j_{\mathrm{capacity}}} \tag{1}$$

令 be_j 为 t_i 在各虚拟机上执行的起始时间，ECT_{ij} 为任务 t_i 在虚拟机 vm_j 上执行的期望完成时间，则其可表示为

$$\mathrm{ECT}_{ij}=be_j+\mathrm{ETC}_{ij} \tag{2}$$

所有任务的完成时间表示为 Makespan，则

$$\mathrm{Makespan}=\max\{\mathrm{ECT}_{ij}\}$$

任务调度的目标函数与约束条件如式（3）所示。

$$\min\{\mathrm{Makespan}\}$$
$$\begin{cases} t^i_{\mathrm{memory}}\leqslant vm^j_{\mathrm{memory}} & i=1,2,\cdots,n \\ t^i_{\mathrm{bw}}\leqslant vm^j_{\mathrm{bw}} & j=1,2,\cdots,m \end{cases} \tag{3}$$

2.2　动态优先级模型

任务的优先级反映了任务的重要程度，其设定要能够兼顾公平和效率，满足用户对费用、截止时间等 QoS 需求。为此，本文综合考虑任务价值密度和执行紧迫性，构建了一种动态优先级。

1）任务的价值密度

由于单纯的任务价值并不能反映出任务的实际价值，为了对比不同长度任务的价值，本文用任务单位大小获得的用户期望费用来表示任务价值密度（task value density，TVD），表示为

$$\mathrm{TVD}_i=\frac{t^i_{\mathrm{fee}}}{t^i_{\mathrm{mi}}} \tag{4}$$

2）任务执行的紧迫性

为了尽可能地满足任务的时间约束，提高任务的成功执行率，本文以任务的等待时间与任务的剩余时间的加 1 求和的比来表示任务执行的紧迫性（urgency of the task execution，UTE），表示为

$$\mathrm{UTE}_i=\frac{t^i_{\mathrm{wait}}}{1+t^i_{\mathrm{left}}} \tag{5}$$

其中：t^i_{wait} 为任务等待的时间，$t^i_{\mathrm{wait}}=t_{\mathrm{current}}-t^i_{\mathrm{submit}}$，$t_{\mathrm{current}}$ 为当前时间；t^i_{left} 为任务的剩余时间，$t^i_{\mathrm{left}}=t^i_{\mathrm{submit}}+t^i_{\mathrm{deadline}}-t_{\mathrm{current}}$，$t^i_{\mathrm{left}}\geqslant 0$。显然，随着等待时间 t^i_{wait} 的增加，空闲时间 t^i_{left} 相应地会减小，UTE_i 会迅速增加，优先级的动态特性由此体现出来。

3）任务的动态优先级

为了构建动态优先级，需要对相关的 TVD_i、UTE_i 进行标准化处理。本文采用基于原始数据的均值和标准差的 Z-score 方法进行数据标准化。以 st_{ik} 表示任务 t_i 的第 k 个优先级指标标准化结果，其计算公式为

$$st_{ik}=(z_{ik}-\overline{z_k})/\delta_k \tag{6}$$

其中：$k=1,2$；z_{ik} 属于矩阵 $Z=\begin{Bmatrix} z_{11} & z_{12} \\ \vdots & \vdots \\ z_{n1} & z_{n2} \end{Bmatrix}=\begin{Bmatrix} \mathrm{TVD}_1 & \mathrm{UTE}_1 \\ \vdots & \vdots \\ \mathrm{TVD}_n & \mathrm{UTE}_n \end{Bmatrix}$；$\overline{z_k}$ 是矩阵 Z 第 k 列的平均值，$\overline{z_k}=\frac{1}{n}\sum_{i=1}^{n}z_{ik}$；$\delta_k$ 是所有任务第 k 个指标的标准差，$\delta_k=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(z_{ik}-\overline{z_k})}$。

用 $DP(t_i)$ 表示任务 t_i 的动态优先级，则采用加权的方式计算任务动态优先级的公式为

$$DP(t_i)=\omega_1\times st_{i1}+\omega_2\times st_{i2} \tag{7}$$

其中：$\omega_1,\omega_2\in[0,1]$ 是权重因子，且满足 $\omega_1+\omega_2=1$。

3　TS-PFB 算法

本文综合考虑任务的价值密度和任务执行的紧迫性构建了任务的优先级，为了在减少整体任务调度时间的同时维持虚拟机的负载均衡，本文模仿萤火虫行为，设计了任务调度的决策变量，任务调度流程框图如图 1 所示。

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

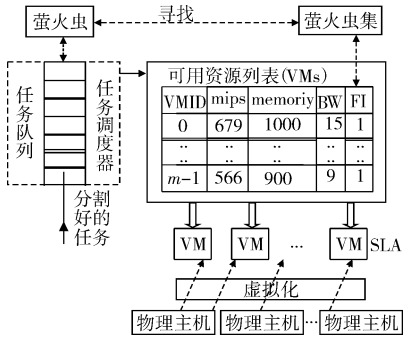

图 1　任务调度流程框图

3.1　萤火虫算法

萤火虫算法(firefly algorithm,FA)[15]是受自然界中的萤火虫通过荧光进行信息交流这种群体行为的启发演变而来的随机优化算法。算法对萤火虫生物发光特性及趋光行为进行了理想化处理:萤火虫不分性别,它将会被吸引到所有其他比它更亮的萤火虫那去;萤火虫的吸引力和亮度成正比,对于任何两只萤火虫,其中一只会向着比它更亮的另一只移动,然而亮度是随着距离的增加而减少的;如果没有找到一个比给定的萤火虫更亮,它会随机移动。

其仿生原理是:把搜索空间中的点模拟为萤火虫个体,将搜索和优化过程模拟成萤火虫个体的吸引和移动过程,将求解问题的目标函数度量成个体所处位置的优劣,将个体的优胜劣汰过程类比为搜索和优化过程中用好的可行解取代较差可行解的迭代过程[15]。

3.2　模仿萤火虫行为的决策变量

在 FA 算法中,萤火虫彼此吸引的原因取决于两个要素,即自身亮度和吸引度。而任务调度时,任务和虚拟机也是一个相互吸引的过程。在该过程中,可以将所需要考虑的期望最早完成时间和虚拟机的负载情况分别作为吸引度和亮度,模拟萤火虫亮度吸引的行为对任务进行调度。

吸引度(attractiveness):期望最早完成时间 ECT_{ij},其值越小,吸引度越大。

荧光亮度(fluorescence intensity,FI):初始时刻,每个虚拟机亮度最亮,值为 1,虚拟机负载增加时,亮度降低,以每个虚拟机上负载的期望处理时间(EPT_j)与初值 1 的和来作为萤火虫的荧光亮度。

若分配给虚拟机 vm_j 的任务的总长度称为负载,记为 L_j,则 $\mathrm{EPT}_j = \dfrac{L_j}{vm^j_{\mathrm{capacity}}}$。

vm_j 的荧光亮度计算公式如式(8)所示。

$$FI_j = 1 + \mathrm{EPT}_j \tag{8}$$

最后综合考虑吸引度和荧光亮度的决策变量 ρ_{ij},如式(9)所示。

$$\rho_{ij} = \frac{1}{\mathrm{ECT}_{ij}} \times \frac{1}{FI_j} \tag{9}$$

显然,ECT_{ij} 和 FI_j 越小时,ρ_{ij} 越大。将任务 t_i 调度到可行虚拟机组中最大 ρ_{ij} 值所对应的 vm 上即可。

4　TS-PFB 算法描述

云计算中 TS-PFB 算法执行步骤如下:首先计算出所有任务的优先级,依据任务的优先级和资源的性能,分别对任务和 VM 资源排序;再按优先级高低依次调度任务,计算出任务在各 VM 上的最早完成时间 ECT,依据任务带宽需求($t^i_{\mathrm{bw}} \leqslant vm^j_{\mathrm{bw}}$)和内存($t^i_{\mathrm{memory}} \leqslant vm^j_{\mathrm{memory}}$)的要求,建立任务与可行 VM 组的关联;在计算出 ETC_{ij} 和虚拟机当前的荧光亮度 FI_j 的基础上,计算任务与可行 VM 组内各 vm 的调度决策变量 ρ_{ij};将任务调度到最大 ρ_{ij} 值所对应的虚拟机上;更新虚拟机的起始执行时间 be_j 与荧光亮度 FI_j,删除已调度任务;如上开始下一个任务调度的调度,直至所有任务完成。

具体算法流程如下:

```
Notations: 令 CT_i 为任务 t_i 的可行 VM 组
Procedure: Init( )          //初始化起始执行时间和荧光亮度
  for( int j = 0; j < m; j ++ )
  be_j = 0; FI_j = 1;
Procedure: DP( t_i )        //计算任务的动态优先级
  for t_i in T
    set DP( t_i ) based on formula( 7 ) ;
Procedure: Sort( T,VM)      //降序排列任务和 VM
  Sort T by DP( t_i ) in descending order;
  Sort VM by vm^j_capacity in descending order;
Procedure: GetETC( )        //计算 ETC_ij
  for t_i in sorted T
    for vm_j in sorted VM
      set ETC_ij based on formula( 1 ) ;
Procedure: Schedule( )
  for t_i in sorted T
    for vm_j in sorted VM       //计算 ECT_ij 和 CT_i
      set ECT_ij based on formula( 2 ) ;
      if( ECT_ij ≤ t^i_deadline & & t^i_bw ≤ vm^j_bw )
      CT_i ← vm_j;
    for vm_j in CT_i            //计算出 ρ_ij,开始调度
      set ρ_ij based on formula( 9 ) ;;
    assign t_i to the vm in CT_i with the max{ρ_ij} ;
    update be_j and FI_j;
end;
```

5　实验仿真与结果分析

5.1　实验环境

为了检验 TS-PFB 算法的性能,本文在 CloudSim 仿真平台上进行了仿真实验。仿真开始时,用户的任务提交给代理后,代理依据在信息服务中心(CIS)获取的注册的数据中心信息,开始了如图 2 所示的一系列仿真步骤。

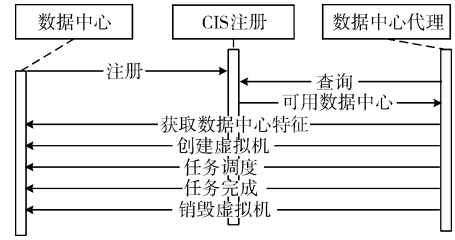

图 2　CloudSim 仿真流程

在仿真实验中,模拟了 50 ~ 450 个独立任务在 20 个 VM 资源上的调度,每个虚拟机含有一台处理器,处理能力范围为[500,1000],存储能力范围为[1000,2000],带宽范围为[10,20],VM 到 host 的分配由 Cloudsim 自带的 Time-Shared 算法实现。任务长度区间为[10000,40000],存储能力需求范围为[800,1800],带宽需求范围为[8,15],截止时间范围为[50,800],优先级权重因子设置为 $\omega_1 = \omega_2 = 0.5$。实验运行在 AMD X2215 2.7 GHz,2.00 GB 的 PC 机上。

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

5.2　结果分析

在相同的测试条件和环境下，本文分别实现了 Min-Min、Max-Min、HBB-LB 和 TS-PFB 算法，并在各类算法总任务平均完成时间 Makespan、负载均衡程度和截止期内成功完成的任务率上进行了对比测试。

实验 1　总任务平均完成时间 Makespan 测试

图 3 描述了以上四种算法在多组不同数目任务量下平均完成时间 Makespan 的对比。从图中可以看出，在不同任务量下，TS-PFB 算法的平均 Makespan 略优于 Max-Min 算法，略长于 HBB-LB 算法（HBB-LB 算法的平均 Makespan 没有考虑任务实际调度过程中任务迁移的时耗），这说明 TS-PFB 算法能够有效地减小总任务的完成时间。

实验 2　负载均衡程度测试

令分配给虚拟机 vm_j 的任务的总长度称为负载，记为 L_j。虚拟机 vm_j 上载有任务的期望处理时间 $\mathrm{EPT}_j = \frac{L_j}{vm^j_{\mathrm{capacity}}}$。所有虚拟机的期望处理时间的平均值为 $\overline{\mathrm{EPT}} = \frac{1}{m}\sum_{j=1}^{m}\mathrm{EPT}_j$。本文用期望处理时间的标准差来判断虚拟机群的负载均衡程度，如式（10）所示。

$$\sigma = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(\mathrm{EPT}_j - \overline{\mathrm{EPT}})^2} \tag{10}$$

依据式（10）计算出四种算法的平均负载均衡程度测试结果，如图 4 所示。从图 4 中可以看出，在四种算法中，Min-Min 算法则出现了严重的负载失衡；动态负载均衡算法 HBB-LB 设定阈值为 30，可见触动负载均衡策略，之后 σ 值趋于平稳；而 TS-PFB 算法调度完后的 *VM* 负载均衡程度 σ 最低，这是因为在任务调度时考虑了 *VM* 的负载约束（荧光亮度）。该实验结果说明 TS-PFB 算法具有很好的负载均衡特性。

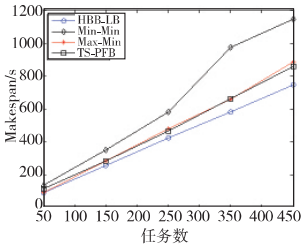

图 3　总任务平均完成时间

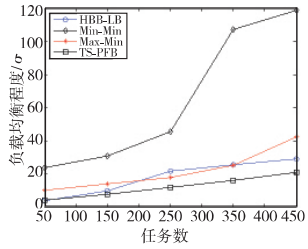

图 4　平均负载均衡程度

实验 3　截止期内任务完成的成功率

令截止期内成功完成的任务数量比上总任务的数量为成功率。四类算法的平均成功率测试结果如图 5 所示。

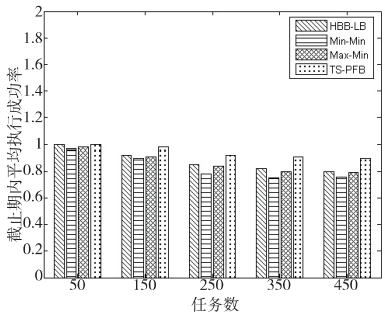

图 5　平均成功率

从图 5 中可以看出，TS-PFB 算法截止期内任务完成的平均成功率最高，即截止期内错失率最小，这是因为任务优先级构建是考虑了任务执行的紧迫性，使得截止期短的任务优先得到调度。

6　结束语

本文提出了一种云计算环境下基于动态优先级和萤火虫行为的任务调度算法。实验结果表明，该算法能够有效地减小总任务的完成时间，具有较好的负载均衡性，也能够保证大部分任务在任务的截止期内完成。以后的工作中将进一步考虑用户安全方面 QoS 需求和更为复杂的任务的调度等问题。

参考文献：

[1] WEISS A. Computing in the clouds [J]. NetWorker, 2007, 11(4): 16-25.

[2] MISHRA A, JAIN R, DURRESI A. Cloud computing: networking and communication challenges [J]. IEEE Communications Magazine, 2012, 50(9): 24-25.

[3] GHANBARI S, OTHMAN M. A priority based job scheduling algorithm in cloud computing [J]. Procedia Engineering, 2012, 50: 778-785.

[4] GU Lian-jun, TANG Zhuo, XIE Guo-qi. The implementation of Map-Reduce scheduling algorithm based on priority [C] //Proc of Parallel Computational Fluid Dynamics. Berlin: Springer, 2014: 100-111.

[5] LIU Gang, LI Jing, XU Jian-chao. An improved Min-Min algorithm in cloud comp uting [C] //Proc of International Conference of Modern Computer Science and Applications. Berlin: Springer, 2013: 47-52.

[6] LI Qi, BA Wei. A group priority earliest deadline first scheduling algorithm [J]. Frontiers of Computer Science, 2012, 6(5): 560-567.

[7] 夏家莉，陈辉，杨兵. 一种动态优先级实时任务调度算法 [J]. 计算机学报，2012，35(12)：2685-2695.

[8] MISHRA M, DAS A, KULKARNI P, *et al*. Dynamic resource management using virtual machine migrations [J]. IEEE Communications Magazine, 2012, 50(9): 34-40.

[9] REN Xiao-na, LIN Rong-heng, ZOU Hua. A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast [C] //Proc of IEEE International Conference on Cloud Computing and Intelligence Systems. Piscataway: IEEE Press, 2011: 220-224.

[10] HU Jin-hua, GU Jian-hua, SUN Guo-fei, *et al*. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment [C] //Proc of the 3rd International Symposium on Parallel Architectures, Algorithms and Programming. Piscataway: IEEE Press, 2010: 89-96.

[11] BABU L D D, KIRSHNA P V. Honey bee behavior inspired load balancing of tasks in cloud computing environments [J]. Applied Soft Computing, 2013, 13(5): 2292-2303.

[12] LI Kun, XU Gao-chao, ZHAO Guang-yu, *et al*. Cloud task scheduling based on load balancing ant colony optimization [C] //Proc of the 6th Annual Chinagrid Conference. [S. l.]: IEEE Press, 2011: 3-9.

[13] MAO Ying-chi, CHEN Xi, LI Xie. Max-Min task scheduling algorithm for load balance in cloud computing [C] //Proc of International Conference on Computer Science and Information Technology. [S. l.]: Springer, 2014: 457-465.

[14] CHEN Huan-kai, WANG F, HELIAN N, *et al*. User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing [C] //Proc of National Conference on Parallel Computing Technologies. Piscataway: IEEE Press, 2013: 1-8.

[15] YANG Xin-she. Firefly algorithm [C] //Proc of Engineering Optimization. 2010: 221-23.

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net