云计算下的基于萤火虫-遗传算法的资源调度①

单好民

(浙江邮电职业技术学院, 绍兴 312000)

摘 要: 如何能够最大限度发挥云计算中资源调度效率是目前研究的热点之一. 首先建立云计算环境下的资源调度模型,将萤火虫算法中的个体与云计算节点资源进行对应,其次在算法中个体初始化中引入遗传算法优化初始解,对算法中的位置更新设定感觉阀值用来调节个体选择最优路径的概率;最后针对挥发因子的改进使得荧光素的值进行更新. 仿真实验表明,该算法能够有效的提高云计算中的资源调度性能,缩短了任务完成的时间,提高系统整体处理能力.

关键词: 云计算; 萤火虫算法; 遗传算法; 资源调度

Resource Scheduling Based on Firefly-Genetic Algorithm in Cloud Computing

SHAN Hao-Min

(Zhejiang Technical College of Posts & Telecom, Shaoxing 312000, China)

Abstract: How to give the fullest play to the efficiency of resource scheduling in cloud computing is a hot spot of current research. First of all, resource scheduling model in cloud computing is established and individuals in firefly algorithm and node resources in cloud computing are matched; secondly, the genetic algorithm is introduced into the initialization of individuals in the algorithm and sensory threshold of the updating of algorithm's position is set to adjust the probability for individuals to choose the optimal path; finally, the volatile factor is improved to update the value of fluorescein. Simulation experiment shows that this algorithm can effectively improve the performance of resource scheduling in cloud computing, shorten the time to complete tasks and improve the system's overall processing capacity.

Key words: cloud computing; firefly algorithm; genetic algorithm; resource scheduling

引言

近年来, 云计算是集分布式计算、虚拟化技术、网格计算和 Web 服务等技术上发展起来的一种综合技术[1][2], 它通过互联网将处于网络中的各个节点的资源进行共享. 但如何能够最大限度的合理利用资源调度则是云计算研究的热门方向.

目前, 国内外学者在云计算环境下的资源调度方面已经进行了大量研究工作, 文献[3]提出了在云计算中的使用一种基于改进粒子群优化算法, 利用 PSO 较快的收敛速度找到云资源调度问题的最优解, 并根据每个粒子的适应度值自适应地改变每个粒子的速度权重,提高了全局寻优能力和收敛能力, 仿真结果表明减少了任务的平均完成时间, 提高了任务处理的效率, 但粒子优化需要一定的时间; 文献[4]提出在整合虚拟云服务资源的基础上, 将模糊集理论引入到基于服务质量(QoS)的云服务资源选择中,取得了一定的效果, 但计算量大, 持续时间长; 文献[5]采用信息熵理论来维护非支配解集, 以保持解的多样性和分布均匀性; 在利用 Sigma 方法实现快速收敛的基础上, 引入混沌扰动机制, 以提高种群多样性和算法全局寻优能力, 避免算法陷入局部最优, 取得了比较好的效果, 但增加了算法的空间复杂性. 文献[6]提出了一种实现云计算负载均衡的双向蚁群优化算法(BACO)用于资源调度, 该算法考虑到了每个虚拟机的负载和计算能力,

① 基金项目:浙江省教育厅科研项目(201432433)
收稿时间:2015-10-22;收到修改稿时间:2015-12-15

同时在云环境中引入了蚂蚁的向前移动和向后移动, 结果表明该算法的总任务完成时间较短, 具有较好的寻优能力, 并且能够实现负载均衡,是一种有效的资源调度算法, 但增加了额外的硬件消耗; 文献[7]提出一种基于膜计算的蝙蝠算法, 将膜系统内部分解为主膜和辅助膜, 在辅助膜内进行蝙蝠的个体局部寻优, 将优化后的个体传送到主膜间进行全局优化, 从而达到了云计算资源优化分配要求, 实验表明算法提高了云计算环境下的系统处理时间和效率, 使得云计算环境下的资源分配更加合理, 其缺点是算法寻优过程复杂; 文献[8]提出一种面向应用性能的云计算弹性资源调整方法. 该方法利用自动伸缩算法, 在垂直层次上对负载需求的波动进行虚拟机资源调整, 以实现动态调整分配资源量来满足应用的服务级别的需求, 优化云计算资源利用率, 其缺点是容易受到需求波动影响; 文献[9]提出基于 Q 学习和双向 ACO 算法的云计算任务资源分配模型, 将其获得的最优策略对应的 Q 值初始化网络中节点的 Q 值, 算法能够实现任务资源的最终分配; 文献[10]提出一种低负载和低成本的资源分配策略, 实现系统负载均衡. 实验结果表明, 该策略在满足 QoS 约束的条件下, 有效地提高资源利用率, 其缺点是负载均衡无法控制; 文献[11]提出在云计算资源算法中引入安全机制, 保证资源传输过程中资源丢失; 文献[12]提出了一种结合请求率预测与能耗感知的弹性资源管理方法,以达到对系统能耗的弹性管理, 实验验证了该方法在保证服务质量的前提下能更有效地降低能耗, 其缺点是应用实践不足; 文献[13]提出从应用提供者收益角度考虑,兼顾 SLA 收益损失和服务器租用成本投入, 提出虚拟机资源调度方法, 旨在使得应用租用者收益最大化, 实验证明取得了不错的效果, 其缺点是没有硬件损耗.

本文针对萤火虫算法在资源调度方面的不足的基础上, 引入了遗传算法进行改进,使得改进后的算法在处理并行任务的资源分配方面有了进一步的提高,通过实验进一步证明了本文算法在云计算资源分配方面具有一定的优越性.

1　云计算下的算法描述

云资源中的资源调度通过如下模型来表示:

$$\begin{cases} \max \sum_i f_i(y_i) \\ s.t. \sum_j x_{ij} = y_i \\ \sum_i x_{ij} \le C_j \\ \dfrac{I_i}{D_i} \le y_i \\ x_{ij} \ge 0 \\ \min \sum_{i,j} C(i,j) \\ \max \sum_{i,j} E(i,j) \\ \min \sum_{i,j} S(i,j) \end{cases} \tag{1}$$

在公式(1)中, C_j 表示虚拟机 j 的最大计算处理能力, I_i 表示作业 i 包含的机器指令条数, D_i 表示任务最迟完工时间, y_i 表示分配给任务 i 的资源总和,表示为 $y_i = \sum_j x_{ij}$,其中 $\sum_j x_{ij}$ 是表示虚拟机 j 占用计算资源总和. $f_i(y_i)$ 表示为目标函数中效率函数, 根据云计算的资源调度的要求,云计算资源调度的最大化为 $\max \sum_i f_i(y_i)$.

将云计算中的资源与萤火虫个体进行对应, 通过进一步优化萤火虫算法找到最有个体, 从而能够找到云计算中的资源分配方案.

1.1 人工萤火虫算法

假设萤火虫的群体为 N , 第 i 只萤火虫所在的位置为 (x_i, y_i) , 第 i 只萤火虫所对应的目标函数为 $f(x_i, y_i)$, 第 i 只萤火虫的萤光素的值为 T_i , $x_j(t)$ 表示第 t 代的第 j 个萤火虫的位置, $l_j(t)$ 表示第 t 代的第 j 个萤光素的值, 萤火虫的视野范围更新如下:

$$R_d^j(t) = \max\{0, R_d^{j-1}(t) + \beta(|N_i(t)|)\} \tag{2}$$

$$N_i(t) = \{\|x_j(t-1) - x_i(t-1)\| < R_d^i\} \tag{3}$$

萤火虫个体位置更新公式:

$$x_i(t) = x_i(t-1) + \frac{x_i(t-1) - x_{i-1}(t-1)}{\|x_i(t-1) - x_{i-1}(t-1)\|} \tag{4}$$

荧光素值的公式:

$$l_i(t+1) = (1-\rho) l_i(t) + \lambda J(x_i(t+1)) \tag{5}$$

1.2 遗传算法

该算法是模拟遗传选择和自然淘汰的一种随机概率搜素算法. 其算法的组成主要包含 4 个步骤:

(1) 数据编码方案的确定, 随机产生一个初始化个体;

(2) 给出评价个体优劣的适应度值; 其中个体的

评价是对每一个体计算其路径的长度, 并将该长度作为个体的适应度函数, 表示如下:

$$f(x)=\sum_{i=1}^{n-1} length(i,j)+d(1,n) \tag{6}$$

式中, $length(i,j)$ 表示两个个体 i 和 j 之间的距离. 适应度越小的个体表示路径越短,则该个体越好.

(3) 判断算法是否满足收敛的条件, 如果满足则输出搜素结果, 否则继续执行.

(4) 分别按照交叉概率和变异概率来进行执行交叉操作.

2 基于萤火虫-遗传算法在云计算资源中的应用

2.1 个体初始解的优化

采用遗传算法来初始萤火虫个体的初始解,使得萤火虫算法的后期效率得到明显的提高. 设定萤火虫算法的种群大小为 M ,将所有的萤火虫个体分为两个种群, 分别为父种群和子种群. 其中父种群为 $Z=\{\alpha_1,\alpha_2,\ldots a_M\}$,设定每一个个体的适应度函数为 $f(\alpha_i)$,子群 $X=\{\delta_1,\delta_2,\ldots\delta_M\}$

(1) 变异操作: 第 g 代种群中的第 i 个个体 x_i^g 依据公式(4)的变异方式.

$$V_i^{g+1}=x_i^g+F*(x_{r1}^g-x_{r2}^g) \tag{7}$$

式(7)中的 V_i^{g+1} 是变异后的种群中的个体, F 为随机因子, 主要是用来控制差分向量的缩放程度, 设定值为[0,1]之间.

(2) 交叉操作: 通过一定的概率选择, 将变异的中第 i 个体 x_i^g 与父代个体 V_i^{g+1} 之间在第进行交叉, 得到新的个体

$$\kappa_i^g=\begin{cases}V_i^{g+1}, & t\in[0,1]\\ x_i^g, otherwise\end{cases} \tag{8}$$

式(8)中可以保证在交叉过程出现一个 0 到 1 之间的随机整数, 能够保证 $\kappa_{i,j}$ 至少有一个分量来自 $y_{i,j}$.

(3) 选择操作. 在进行选择个体使用“贪婪”选择策略, 使用适应度函数 f 进行比较大小, 选择 f 值大的个体进入下一代中, 即通过变异与交叉操作后生产的新的个体 κ_i^g 与上一代个体 x_i^g 进行比较, 如果小于则保持 x_i^g 不变, 否则直接进入下一代.

通过上述三个步骤, 选取的个体 M_i 为 $b_{\min(i1,i2,\ldots in)}$, 将 M_i 与子群 X 进行合并, 使得 $X(t)=X(t-1)\cup M_i$. 然后将这些个体通过交叉, 变异等操作得到最后解, 作为萤火虫算法的初始解.

2.2 阀值设定-个体的选择

萤火虫个体在进行前进方向选择的策略是根据荧光素的值大小来进行判断, 这就容易导致在某一个方向上容易产生局部最优, 导致后续的萤火虫个体存在以较大的概率集中选择在当前局部最优的前进位置上, 为了避免这种情况的发生,设定一个阀值 γ, 使得设定在路径上的荧光素的值小于阀值的时候,萤火虫个体可以忽略该前进方向的荧光素的值, 因此可以继续寻找. 反之, 萤火虫个体选择选择荧光素上的前进方向.

因此第 k 只萤火虫个体按照以下的概率从状态 i 到状态 j 进行转换, 其中 $allowed_k$ 为状态列表.

$$j=\max\{\tau_i^{\alpha},\eta_i^{\beta}\},s\in allowed_k \quad if\ \lambda<\gamma \tag{9}$$

2.3 针对 ρ 的改进

在公式(5)中, 由于荧光素更新公式中的发挥因子 ρ 的作用, 使得没有在搜索路径上的荧光素的值逐步降低直至不被选择, 与此同时局部一些搜索路径上的发挥因子 ρ 的值逐渐变大的时候, 有效解的信息量逐渐增大导致以前经过搜索的路径又重新参与全局路径的选择, 导致了降低了全局搜素最优能力个体解的降低,消耗了搜素的时间. 因此应该对发挥因子 ρ 的处理就显得尤为重要, 本文采用自适应的因子 η 来改变 ρ 的值, 设定 ρ 的值初始值为 0.99, 最小值为 $\rho_{\min}$, 按照一定的循环次数进行逐渐递减;

$$\rho(t+1)=\begin{cases}0.99-\eta\bullet rand()/100\bullet time & \rho(t)>\rho_{\min}\\ \rho_{\min}\end{cases} \tag{10}$$

式中, $rand()$ 是一个随机函数, 设定 η 为自适应因子, $time$ 为循环次数,这种自适应因子可以保证在一定的搜素范围下的算法的全局搜索能力.

2.4 算法步骤描述

求解步骤如下, 流程如图 1 所示.

Step1. 初始化萤火虫算法的各个参数, 设定种群数目 Num, 算法迭代次数为 N, 最大次数为 max , 满足 $N<\max$, 初始化萤火虫的个体位置, 定义合适的荧光素值;

Step2. 针对云计算资源分配的特点, 将云计算节点的资源与萤火虫的个体位置进行一一对应;

Step3. 根据 3.1 描述对萤火虫算法的个体进行初始化;

Step4. 根据式(2)和(3)计算单个萤火虫的位置和

荧光素的值, 找到各自目前节点的位置和荧光素的值.

Step5. 根据式(4)(5)计算萤火虫最佳的位置, 然后根据式(7)(8)对萤火虫位置进行高斯变异更新, 在目前的位置上得到个体的新位置($x_{new}(t), x_i(t)$).

Step6. 计算每一个萤火虫的 $x_{new}(t)$, 如果 $x_{new}(t) > x_i(t)$,则 $i = new$,否则再更新.

Step7. 如果迭代次数小于 max , 转到 Step8.

Step8. 如果满足终止条件, 寻优过程便结束, 否则转向 Step4 继续优化.

Step9. 根据萤火虫中的最优个体的位置变换为相应的云计算资源分配方案.

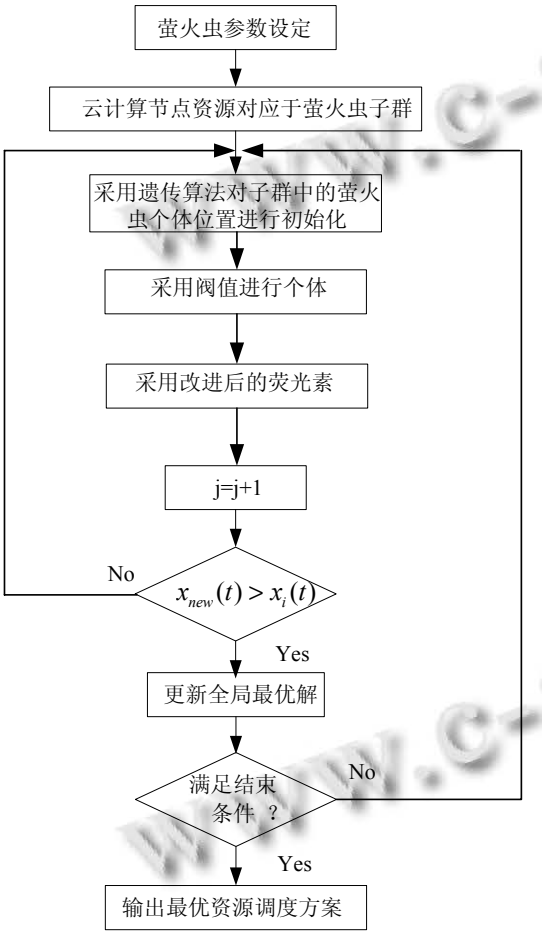

图 1　基于萤火虫-遗传算法的云计算资源调度

3　算法仿真

为了进一步验证本文算法在云计算资源中调度的优越性, 本文从 2 个方面来进行比较:(1)与近几年发表的有关代表性的参考进行比较;(2)与经典的智能算法进行比较. 本文采用 CloudSim[14]平台进行测试, 选择 CPU 为酷睿 i3 和 4GDDR3, Windows Xp, 仿真软件采用 matlab2012 进行模拟.

3.1 与最近文献算法进行比较

设定虚拟任务为 400 个,虚拟节点为 100 个, 设置迭代次数为 300, 将本文的基于萤火虫-遗传算法和文献[3]和文献[10]的算法在云计算模型中不同任务数下进行比较.

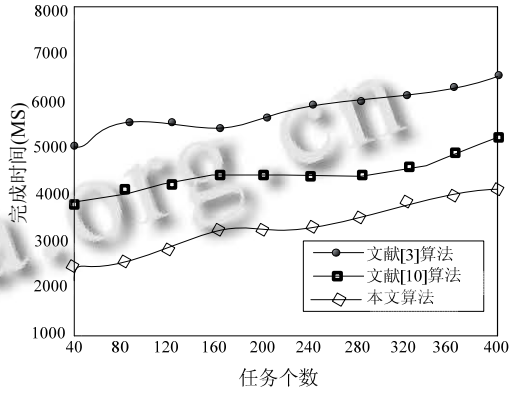

图 2　3 种资源负载算法任务完成时间比较

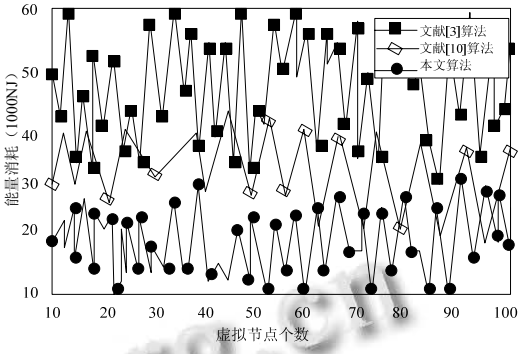

图 3　3 种资源负载算法能量消耗时间比较

3.2 与经典的智能算法进行比较

本文选取了云计算下的三种不同的虚拟任务情况下的算法运行效果对比. 分别选取了任务数为 1000,任务数为 10 万和任务数为 100 万作为研究对象.

(1) 虚拟任务为 1000 的算法性能对比

采用遗传算法、萤火算法、本文算法对任务为 1000 和资源数量为 100 进行分配, 比较结果如图 4 所示. 从图 4 可以发现任务数量在 400 左右成为了分界点, 完成之前三种算法相差不大. 分界点之后三种算法之间才出现区别, 基本上满足云计算条件下的资源分布.

(2) 虚拟任务为 10 万的算法性能对比

采用遗传算法、萤火算法、本文算法对任务为 10 万和资源数量为 1000 进行分配, 比较结果如图 5 所示. 从图 5 可以发现任务数量开始的阶段就区分开来, 并

且本文算法与萤火虫算法在开始阶段相差不大, 伴随着任务数量逐渐增大, 三种算法呈现比较大的区别, 本文的算法适合在云计算条件下的虚拟任务较大的资源调度问题.

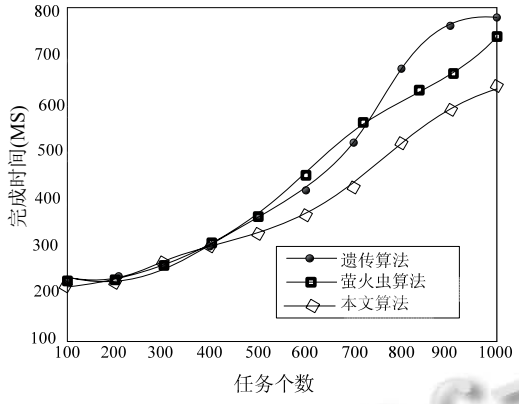

图 4 虚拟任务为 1000 的算法性能对比

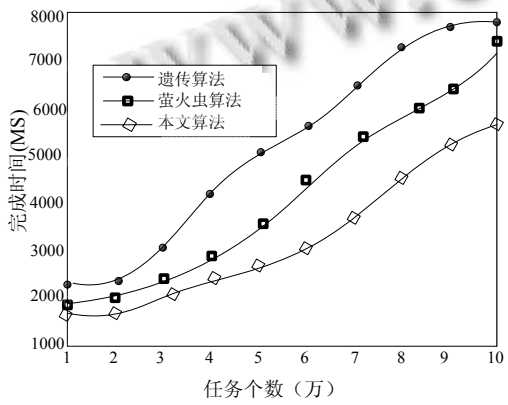

图 5 虚拟任务为 10 万的算法性能对比

(3) 虚拟任务为 100 万的算法性能对比

采用遗传算法、萤火算法、本文算法对任务为 100 万和资源数量为 1000 进行分配, 比较结果如图 6 所示. 从图 6 可知, 随着虚拟任务数增加, 当任务数达到 50 万的时候, 本文的算法的任务完成时间要远远少于其他, 主要是原因是因为算法中进行了个体初始化以及采用了阀值来设定和针对进行了改进. 本文的算法适合在非常适合云计算条件下的虚拟任务非常大的资源调度问题, 效果越明显.

4 结束语

通过建立云计算环境下的资源调度模型, 将萤火虫算法中的个体与云计算节点资源进行对应, 其次在算法中个体初始化中引入遗传算法优化初始解, 对算法中的位置更新设定感觉阀值用来调节个体选择最优路径的概率;最后针对挥发因子的改进使得荧光素的值进行更新. 通过实验证明, 本文算法能够有效满足云计算资源分配模型, 为云计算资源分配提供了一定的参考价值.

参考文献

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

2 葛新.基于云计算集群扩展中的调度问题研究[学位论文].合肥:中国科学技术大学,2011.

3 蔡琪,单冬红,赵伟艇.改进粒子群算法的云计算环境资源优化调度.辽宁工程技术大学学报(自然科学版),2015,35(1):93−96.

4 Wang S, Dey S. Rendering adaptation to address communication and computation constraints in cloud mobile gaming. 2010 IEEE Global Telecommunications Conference (GLOBECOM 2014). IEEE. 2010. 1−6.

5 张恒巍,韩继红,卫波,王晋东.基于 Map-Reduce 模型的云资源调度方法研究.计算机科学,2015,42(8):118−123.

6 王常芳,徐文忠.一种用于云计算资源调度的双向蚁群优化算法.计算机测量与控制,2015,23(8):2861−2863.

7 宁彬,谷琼,吴钊,袁磊,胡春阳.基于膜计算的蝙蝠算法在云计算资源调度的研究.计算机应用研究,2015,32(3):b 830−833.

8 申京,吴晨光,郝洋.面向云计算数据中心的弹性资源调整方法.南京理工大学学报(自然科学版),2015,39(1):89−93.

9 孙花,朱锦新.基于Q学习和双向ACO算法的云计算任务资源分配模型设计.计算机测量与控制,2014,22(10): 3343−3347.

10 方义秋,郑剑,葛君伟.一种云环境下基于 QoS 约束的资源分配策略.计算机应用与软件,2015,32(1):34−38.

11 Liang H, Huang D, Cai L X, et al. Resource allocation for security services in mobile cloud Computing. IEEE Conference on Computer Communications Rkshops. IEEE. 2014, 12. 191−195.

12 熊伟,李兵.云计算环境下基于能耗感知的弹性资源管理机制.四川大学学报(工程科学版),2015,47(2):112−116.

13 叶世阳,张文博,钟华.一种面向 SLA 的云计算环境下虚拟资源调度方法.计算机应用与软件,2015,32(4):11−17.

14 Calheiros RN, Ranjan R, Caf de R, Buyya R. CloudSim: A novel framework for modelingand simulation of cloud computing infrastructures and services. arXiv:0903.2525. [2009-9-3].