基于改进蚁群算法和粒子群算法的云计算资源调度①

单好民

(浙江邮电职业技术学院, 绍兴 312000)

摘 要: 针对云计算资源分配中存在分配不均、分配效果不好的问题, 利用改进后的蚁群算法和粒子群算法进行资源分配. 首先针对粒子群算法的惯性权值进行改进, 设定适应度函数并选择最佳位置的粒子, 然后将该粒子的位置转变为蚁群算法的初始信息素的值, 通过狼群算法改进蚁群算法的信息素的选择. 仿真实验表明, 本文算法与蚁群算法、粒子群算法相比在任务完成时间、能量消耗方面都有了明显的改善.

关键词: 资源分配; 惯性权重; 信息素; 蚁群算法; 粒子群算法

Cloud Computing Resource Scheduling Based on Improved Ant Colony Algorithm and Particle Swarm Algorithm

SHAN Hao-Min

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

Abstract: Aiming at uneven resource distribution in cloud computing and bad distribution effect, this paper distributes resources in improved and colony algorithm and particle swarm algorithm. First of all, improve the inertia weight value of particle swarm algorithm, set the fitness function and select particles at the optimal location, then convert the location of selected particles into the value of ant colony algorithm's initial pheromone and improve ant colony algorithm's selection of pheromone through wolves algorithm. Through simulation experiment, compared with ant colony algorithm and particle swarm algorithm, algorithm in this paper has been significantly improved in time to complete tasks and energy consumption.

Key words: resource distribution; inertia weight; pheromone; ant colony algorithm; particle swarm algorithm

1 引言

云计算是目前流行的一种商业服务计算模式, 它通过互联网将分布在异地的计算机按照一定的逻辑关系连接在一起, 从而进行资源的存储和使用[1]. 在云计算中, 处于云端的各个任务根据实际需求来获得相应的资源, 但经常会出现资源分配不均匀、不合理的问题, 导致资源分配无法达到最优状态. 因此, 研究云计算下如何更好地分配资源具有非常重要的意义. 国内外学者从不同的角度提出了研究成果: 文献[2]提出了一种基于模拟退火思想的改进遗传算法, 仿真结果表明具有较好的可行性和实用性; 文献[3]提出了一种基于贪心算法的云计算资源调度策略.该算法使得任务完成时间短、资源利用率高; 文献[4]提出了一种基于时间成本负载加强型的蚁群算法, 仿真实验说明该算法具有良好的效果; 文献[5]提出了在云计算中使用改进的粒子群算法进行云计算资源调度, 该算法能够有效的提高云计算下的资源分配效率, 降低消耗的时间; 文献[6]提出了首先以应用性能、保证云应用的服务质量和提高资源利用率为目标的多目标优化模型, 并结合最新的 RBF 神经网络和改进粒子群算法对其求解; 文献[7]提出在云计算资源算法中引入另一种智能算法-布谷鸟算法, 通过对布谷鸟算法性能的改进, 使得算法收敛精度提高, 提高了云计算环境下的资源分配效率; 文献[8]从如何进行云计算资源分配的角度出

① 基金项目:浙江省教育厅科研项目资助(Y201432433);浙江省教育技术研究规划课题(JB119)
收稿时间:2016-10-23;收到修改稿时间:2016-12-19 [doi:10.15888/j.cnki.csa.005870]

发，提出了对蚁群算法的改进，实验表明改进后的算法能够有效的提高云计算资源利用率.

蚁群算法和粒子群算法都是群体智能算法，在解决资源分配优化问题上都取得了比较好的效果. 其中蚁群算法适合解决离散优化问题，粒子群算法适合解决连续优化问题. 本文首先在云计算环境下采用粒子群算法进行资源搜索，搜索完成后形成一个初始解，然后将初始解转变为蚁群算法的信息素，进而继续搜索并得到资源分配的最优解.

2　云计算资源调度概述

资源调度是将资源分配给不同用户的过程，在分配过程中按照单个或多个优化目标进行，这些目标一般包括任务完成时间、任务花费成本和资源利用率等内容[9]. 由于云计算具有动态性和异构性等特点，使得云计算资源调度变得较为复杂. 因此一个良好的云计算资源调度算法不仅能够提高资源的利用率，降低用户完成任务的时间，提高云计算系统的处理能力，还可以为用户提供更加多样化的服务，满足不同的云节点用户在服务质量方面的需求. 其调度模型如图 1 所示.

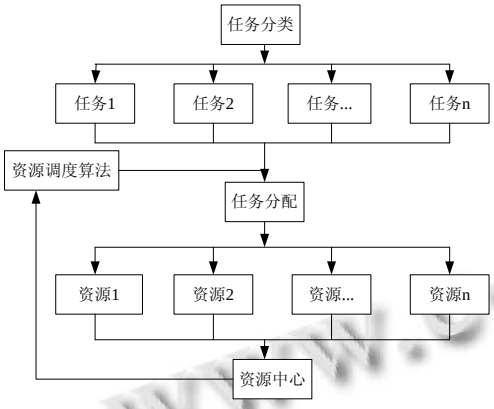

图 1　云计算资源调度模型

云计算资源调度的目标是通过一定的策略来实现云节点上用户的多种需求，提高资源利用率，满足云计算资源效益最大化. 云计算资源调度主要从以下几个方面进行考虑.

(1) 任务调度完成时间最小化：云计算中的任务调度从开始到结束所需要消耗的时间应该尽可能的保持最小化. 从用户的角度看，当用户向云计算系统提交任务后，希望得到云计算系统的及时响应. 因此，完成任务时间最小化是系统需要考虑的.

(2) QOS(服务质量)：云计算系统调度 QOS 一方面包括资源的传输带宽、存储能量和处理速度；另一方面包括用户对于 QOS 中的成本大小、响应时间和可靠性等方面的要求，这样有助于云计算服务商的长远发展.

(3) 负载均衡：衡量云计算系统性能的重要指标之一就是负载均衡，它在一定程度上能够体现算法的优劣. 由于云计算系统各节点的处理能力都不相同，完成任务所需要资源条件也不尽相同，因此，如果要达到云端中的每一个资源节点都能充分利用的效果，就需要将任务均衡地分配到各个资源节点上.

3　基于蚁群算法-粒子群算法的云计算资源调度算法

3.1　云计算下的资源分配模型

云计算下的资源调度就是将 n 个相互独立的任务分配给 m 个资源节点(一般认为，任务个数大于资源分配个数)，资源节点为 $N=\{N_1,N_2,\cdots N_m\}$，其中，N_m 表示第 m 个资源节点. 任务集合为 $T=\{T_1,T_2,\cdots T_n\}$，其中，T_n 表示第 n 个子任务. 将任务和资源之间的关系用矩阵表示，矩阵为:

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{1}$$

式(1)中，a_{mn} 表示资源节点 N_m 和任务节点 T_n 之间的关系，值为 0 或者 1，其中 0 表示没有分配，反之则分配. 因此云计算资源中的节点描述为 $N_i=\{n_{i1},n_{i2},n_{i3}\}$，任务描述为 $T_i=\{t_{i1},t_{i2},t_{i3}\}$，其中 n_{i1} 表示节点 i 的处理能力，n_{i2} 表示节点需要内存，n_{i3} 表示节点带宽；t_{i1} 表示任务需要处理的能力，t_{i2} 表示任务需要的内存，t_{i3} 表示任务所需要带宽.

3.2　粒子群算法的优化

在基本粒子群算法中，粒子个体不断地更新自身的信息来获得最优的状态. 从粒子群算法的公式来看，粒子运动主要是依靠粒子当前的位置、个体最优解和全局最优解来生成新的解. 为了能够更好地提高云计算的资源分配效率，需要对其参数进行改进.

3.2.1　惯性权重

在粒子群算法中，惯性权重 ω 是一个重要的参数，

在局部最优和全局最优中起着重要的平衡作用. 文献[10]指出当 ω 值较大时, 粒子的运动相对强一些, 下一步探索的能力也会变强, 反之, 粒子的开发能力会比较强一些. 因此, 通过对 ω 值的大小进行适当的调整, 可以对全局搜索和局部搜索能力进行权衡. 为了提高算法的灵活性, 增加优秀粒子的产生概率, 在后期搜索中利用随机特性对惯性权重 ω 进行调整.

$$\omega=\begin{cases}\omega_{\max}-\dfrac{iter}{iter_{\max}}\times\omega_{\min} & iter<iter_{pre}\\ randn+\omega_{\min}+(\omega_{\max}-\omega_{\min})\times rand & iter\ge iter_{per}\end{cases} \quad (2)$$

式(2)中, $\omega_{\max}$ 和 $\omega_{\min}$ 分别表示权重最大值和最小值, $iter$ 表示迭代次数, $iter_{pre}$ 表示预定次数, 伴随着迭代次数的不断增大, ω 呈现线性减少, 这样粒子群算法在算法初期具有全局的收敛能力, 收敛速度加快. 而在后期, 为了避免陷入局部最优, 当迭代次数达到预定次数的时候, ω 不再线性减少, 而是采用随机分别惯性权重, $rand$ 为 0 到 1 之间的随机数, $randn$ 为符合正态分布随机数. 采用正态分布的随机数可以保证算法在后期能够跳出局部最优的限制, 这样可以保持多样性, 提升全局搜索能力并保持收敛.

3.2.2 适应度函数

本文将资源调度中的完成时间作为适应度函数的对象, 资源 j 的任务完成时间为 $Time_j=\sum time_{i,j}$, 其中, i 表示资源 j 上执行的任务. 当所有资源完成的时间则为完成的最长时间为 RT, $RT=\max(Time_j)$, 因此, 适应度函数为 $Finess=\dfrac{1}{RT}$.

3.2.3 粒子群算法的位置与速度

云计算资源调度中使用粒子群算法进行搜索的时候, 需要定义算法中的粒子的位置和速度, 粒子的位置在资源调度中代表一个可行的方案, 速度是计算粒子位置变换的变量, 采用矩阵来表示粒子的位置和速度.

$$X_i=\begin{bmatrix}x_{11}\ x_{12}\ \cdots x_{1n}\\ x_{21}\ x_{22}\ \cdots x_{2n}\\ \cdots\ \cdots\ \ \cdots\\ x_{m1}\ x_{m2}\cdots x_{mn}\end{bmatrix} \quad (3)$$

$$V_i=\begin{bmatrix}v_{11}\ v_{12}\ \cdots v_{1n}\\ v_{21}\ v_{22}\ \cdots v_{2n}\\ \cdots\ \cdots\ \ \cdots\\ v_{m1}\ v_{m2}\cdots v_{mn}\end{bmatrix} \quad (4)$$

x_{ij} 表示任务 i 在资源 j 上的取值为 1 或者 0. 因此粒子的位置和速度更新为:

$$x_{ij}(t+1)=\begin{cases}0\\1\end{cases} \quad (5)$$

$$v_{i,j}(t+1)=\omega v_{i,j}(t)+c_1r_1[p_{i,j}(t)-x_{i,j}(t)]+c_2r_2(p_{g,j}(t)-x_{i,j}(t)) \quad (6)$$

式中的有关参数参考基本的粒子群算法的设置, 这里就不做说明了.

3.3 蚁群算法优化

3.3.1 信息素

信息素在蚁群算法中发挥着重要的作用, 直接关系到算法的搜索效果. 如果信息素的初始值设置过大, 算法的初始化迭代就可能失去作用, 反之, 则容易使算法陷入局部最优. 在传统的蚁群算法中, 路径上留下的信息素有最好的或者最坏的情况. 当最好的蚂蚁留下信息素, 则遇到最优路径可能性最大, 否则, 最优路径的可能性最小. 当最坏的信息素留给后续的蚂蚁, 很容易造成算法陷入局部最优. 为了避免这种情况的发生, 引入狼群算法的强者生存原则, 在蚁群算法中找到最优的路径, 并对该路径上的信息素进行释放, 而最坏路径则被放弃, 这样可以保证后续的蚂蚁能够沿着最优的路径前进. 因此信息素更新如下:

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\sum_{k=1}^{m}\Delta\tau_{ij}^{k}+\Delta\tau_{ij}^{*}+\Delta\tau_{ij}^{*'}$$

$$\Delta\tau_{ij}^{*}=\begin{cases}\lambda_1(Q/L^{*})\\0\end{cases} \quad (7)$$

$$\Delta\tau_{ij}^{*'}=\begin{cases}-\lambda_2(Q/L^{*'})\\0\end{cases}$$

L^{*} 为资源上最短的任务长度, $L^{*'}$ 为资源上最长的任务长度, λ_1 和 λ_2 是局部最优和最差的蚂蚁数目. 由于采用了狼群算法的原则, 在信息素进行更新的时候, 信息素之间的差异会逐渐变大, 因此, 需要提前限定信息素在一个阀值范围内. 当超过了设定阀值的最大值, 则选择最大信息素阀值, 反之则选择最小的信息素阀值.

3.3.2 改进的蚁群算法和粒子群算法之间的关联

蚁群算法在算法初期存在信息素匮乏, 容易导致搜索算法效率低下的问题. 通过粒子群算法具有收敛速度快的优点, 将两者算法进行有效的结合. 首先通过粒子群算法进行搜索, 然后将初始解转换为蚁群算法的信息素初始值, 通过蚁群算法完成资源的搜索. 当粒子群算法搜索形成最优解之后, 将最优解转换为信息素分布, 当迭代算法结束后, 将粒子群算法的最

优解添加到集合 S 中, 成为最优解的集合也就是最初的信息素来源.

$$\tau_{ij}^{k} = \tau_0 + \tau_{ij}^{pso} \tag{8}$$

式(8)中, τ_0 是资源的信息, τ_{ij}^{pso} 是粒子群算法最优解转换为信息素的值.

3.4 基于改进的蚁群算法和粒子群算法在云计算中的资源调度

算法流程如图 2 所示.

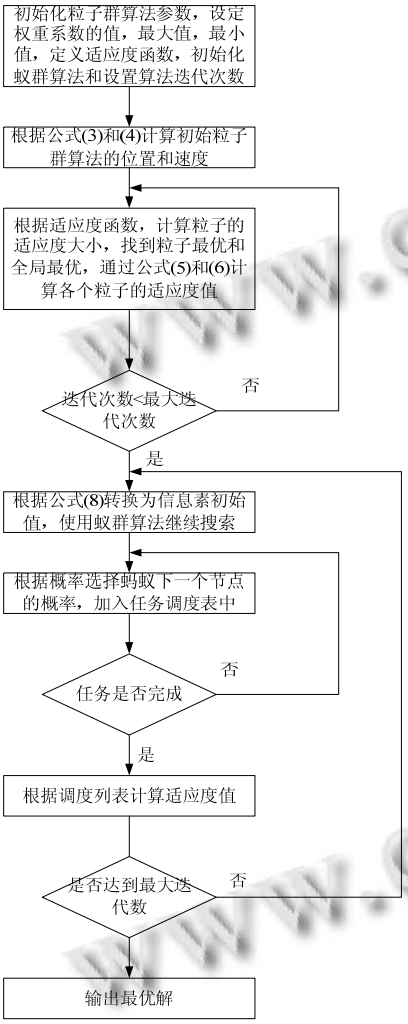

图 2　本文算法流程

图 2 中, 设置蚂蚁个数为 50, 信息素 τ 的值为 0.5, 最大 τ_{max} 为 0.9, τ_{min} 为 0.1, 信息素挥发参数 ρ 为 0.8, 粒子群种群数目 s 为 30, 惯性权重为 0.25, 最小为 0.2, 最大为 0.3 学习因子 c_1, c_2 为 2.迭代次数最大为 100 次.

4　仿真实验

4.1 实验环境

为了进一步说明本文算法在任务-资源上的有效性, 在 CloudSim 平台中假定一个用户提交多次任务, 任务提交的顺序由用户决定, 然后通过资源分配给任务进行执行. 资源的信息主要包括资源编号、资源处理能力、任务数量和资源代价. 表 1 描述了本实验所需要的资源.

表 1　资源信息

资源编号	资源处理能力	任务	资源代价
1	40	2	6
2	30	1	5
3	20	3	7
4	70	4	2
5	70	2	4
6	100	1	8
7	80	4	5
8	90	3	4
9	70	1	7
10	20	2	2

4.2 与基本的蚁群算法, 粒子群算法比较

设置虚拟完成任务 200 个, 图 3 和图 4 描述了三种算法在完成时间、能量消耗方面的比较. 图 3 显示, 与基本蚁群算法、粒子群算法相比, 本文算法在任务完成时间上有了明显减少, 分别减少了约 13.21%和 11.47%. 虽然本文算法的曲线稍微呈现一些波折, 但从整体上看还是满意的. 图 4 显示, 与基本蚁群算法、粒子群算法相比, 本文算法在能量消耗方面低于以上两种算法, 并且图中的曲线相对平缓, 这说明能量消耗稳定, 主要是采用了粒子群算法来为蚁群算法准备信息素, 避免了算法陷入局部最优, 更加容易获得全局最优解, 进一步使得算法趋于稳定. 表 1 记录了每种资源编号对应的处理能力、任务和代价. 表 2 记录了三种算法在 10 组资源中使用情况, 显示三种算法在资源编号为 6, 7, 8 的资源利用率是最高的三组, 而且采用本文算法的资源利用率高于其他两种算法.

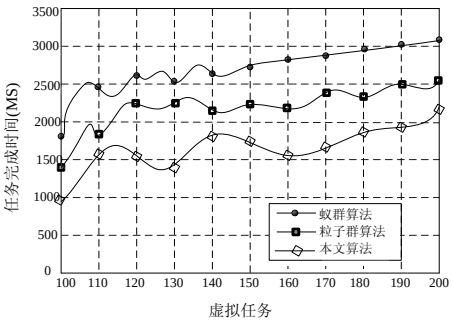

图 3　三种算法完成时间比较

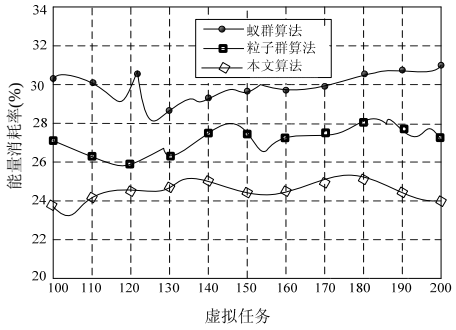

图 4 三种算法在能量消耗方面比较

表 2 10 种资源的利用率

资源编号	蚁群算法(%)	粒子群算法(%)	本文算法(%)
1	23.83	24.78	36.72
2	18.12	21.56	23.61
3	19.52	24.32	28.17
4	67.21	73.72	79.21
5	53.21	62.81	66.72
6	70.28	72.83	80.79
7	63.71	69.62	77.29
8	67.62	70.72	78.91
9	63.18	67.31	71.62
10	47.28	49.82	57.17

4.3 与其他改进的蚁群算法，粒子群算法的比较

设置虚拟完成任务 400 个，图 5 和图 6 显示本文算法与其他改进的算法在任务完成时间、能量消耗方面的比较. 其中文献[4]是改进的蚁群算法，文献[5]是改进的粒子群算法. 图 5 显示，本文算法与文献[4]、文献[5]算法相比， 本文算法在任务完成时间上有了明显减少，分别减少了约 9.31%和 8.47%. 三条曲线都存在一定波折，但本文算法曲线相对平缓，整体上看还是满意的. 图 6 显示，与文献[4]、文献[5]算法相比，本文算法在能量消耗方面要低于以上两种算法，并且图中的曲线相对平缓，这说明能量消耗稳定，主要是采用了粒子群算法为蚁群算法准备信息素. 这说明文献[4]、文献[5]两种算法虽然进行了改进，但总体的改进效果还是低于融合后的算法. 表 3 描述了三种算法在 10 组资源中使用情况，从表中发现三种算法在资源编号为 6, 7, 8 的资源利用率是最高的三组，相比表 2 的利用率分别提高了 4.4%, 3.92%, 3.9%，而且采用本文算法的资源利用率高于其他两种算法.

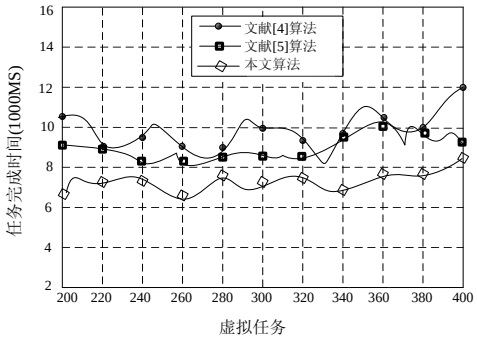

图 5 三种算法的任务完成时间比较

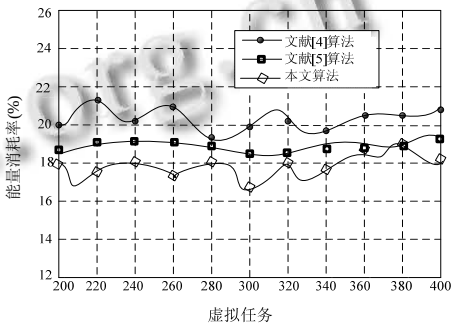

图 6 三种算法在能量消耗方面比较

表 3 10 种资源的利用率

资源编号	文献[4]算法(%)	文献[5]算法(%)	本文算法(%)
1	32.23	37.71	42.12
2	43.32	49.51	55.61
3	31.52	38.12	42.27
4	45.23	52.12	69.34
5	58.23	65.31	75.12
6	75.32	76.43	85.19
7	76.71	77.32	81.21
8	77.62	79.72	82.90
9	72.12	76.31	78.62
10	54.27	55.32	67.12

5 结语

如何更好的进行资源调度是云计算的研究热点之一. 本文将蚁群算法和粒子群算法分别进行改进，并将改进后的算法进行融合，使得融合后的算法的性能有了明显的提高. 仿真实验表明，与蚁群算法、粒子群算法相比，本文算法具有更好的资源调度效率.

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 刘峰,毕利,杨军.一种用于云计算资源调度的改进遗传算法.计算机测量与控制,2016,24(5):202–206.

3 崔雪娇,曾成,徐占然,刘娜.基于贪心算法的云计算资源调度策略.微电子学与计算机,2016,33(6):41–43.

4 聂清彬,蔡婷,王宁.改进的蚁群算法在云计算资源调度中的应用.计算机工程与设计,2016,37(8):2016–2020.

5 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),2016,35(1):93–96.

6 赵宏伟,李圣普.基于粒子群算法和RBF神经网络的云计算资源调度方法研究.计算机科学,2016,43(3):113–117.

7 叶华乔,丁善婷.基于改进的布谷鸟算法在云计算资源的研究.计算机测量与控制,2014,22(12):4150–4154.

8 Chen X. Research of resource sheduling based on ACA-GA in the cloud computing.International Jouranl of Grid and Distributed Computing,2016,9(6):1–12.

9 王艳平.基于蚁群算法的云计算资源调度研究[硕士学位论文].曲阜:曲阜师范大学,2015.

10 姚灿中,杨建梅.基于变惯性权重及动态领域的改进 PSO 算法.计算机工程,2011,37(21):20–22.