Computer Science and Application 计算机科学与应用, 2012, 2, 227-231
http://dx.doi.org/10.12677/csa.2012.24040 Published Online October 2012 (http://www.hanspub.org/journal/csa.html)

Task Scheduling in Cloud Environment Based on Improved PSO Algorithm*

Tingwei Chen, Yi Wei#

Department of Information, Liaoning University, Shenyang
Email: #Weiyi_811@126.com

Received: Sep. 6th, 2012; revised: Sep. 27th, 2012; accepted: Oct. 7th, 2012

Abstract: In a cloud environment, how to map between tasks and resources efficiently has been a hot research topic. Bionic algorithm is widely used in task scheduling optimization, particle swarm optimization algorithm has been paid more attention by virtue of its simple structure, less parameters and easy to achieve. Standard particle swarm optimization algorithm, however, there are more local extreme in cloud computing environment, is prone to make the problem of "premature", which is due to the lack of population diversity. Therefore, it is very important to maintain the diversity of the population in the search process, the paper put forward the idea of a "separatist", the experiments show that the method can solve the problem of "premature" well, it is effective to maintain the diversity of the population, and reduce the completion time of the task effectively.

Keywords: Cloud Computing; Task Scheduling; Particle Swarm Optimization; Division; Precocious

云环境下基于改进 PSO 算法的任务调度策略*

陈廷伟，韦　祎#

辽宁大学信息学院，沈阳
Email: #Weiyi_811@126.com

收稿日期：2012 年 9 月 6 日；修回日期：2012 年 9 月 27 日；录用日期：2012 年 10 月 7 日

摘　要：在云环境中，如何高效的实现任务与资源之间的映射一直是研究的热点问题。仿生算法被广泛的应用于任务调度的优化，其中粒子群优化算法凭借其结构简单、参数少和易实现的优点而受到大力重视。但是，标准的粒子群优化算法对于存在较多局部极值云计算环境，很容易出现“早熟”的问题，这是由于种群多样性的匮乏所致。因此，在搜寻过程中能够保持种群的多样性十分重要，本文提出了“分裂”的思想，经过实验证明，该方法能够很好的解决“早熟”的问题，有效的保持种群多样性，并能有效的减少任务的完成时间。

关键词：云计算；任务调度；粒子群优化算法；分裂；早熟

1. 引言

云计算作为一种新兴的并行计算技术，是分布式处理、并行处理、网格计算的发展和延伸，是这些计算机科学概念的商业实现，适合当今巨型信息化处理需求[1]。云计算提供了更可靠、更安全的存储和计算数据能力、简化计算交付、降低成本、具有更高的扩展性和灵活性。

云计算环境与网格计算环境还有着较大的差别，云的规模较大，其云计算技术的硬件基础设施构建在大规模的廉价服务器集群之上，所以各个节点的计算能力稍低。如何利用有限的资源为用户提供高质量的服务，是云计算需要解决的重要问题。因此，任务调度是云计算的一个重要组成部分。如何高效且快速实现任务与资源之间的映射是研究的重点所在。

*基金项目：辽宁省教育厅科学研究一般项目(L2011004)。
#通讯作者。

2. 改进的 PSO 算法

粒子群优化算法(Particle Swarm Optimization，简称 PSO)最早是由 Eber hart 和 Kennedy 模拟自然界的生物群体觅食而提出的一种优化方法[2]。PSO 算法是一种基于群体的自适应搜索优化算法，其核心思想是群体中个体之间的信息共享有助于整体进化[3]。

在算法中，这些称为“群(Swarm)”的无体积无质量的小粒子都具有记忆的能力，能对自身位置和记忆中经历过的最佳位置进行调整，进而能够调整自身的运动轨迹，同时能够朝着自己以前经历过的最佳位置和整个群体粒子曾经经历过的最佳位置飞行，通过迭代越来越接近最佳位置，最终得到最优解。它具有参数少、易实现的优点，但是，粒子群优化算法也具有自身的不足，该算法在优化过程中容易出现早熟的问题，这主要是由于在粒子群优化的后期由于各个粒子的速度更新能力不足，使得粒子在一定位置紧密聚集而无法进行更大程度、更细致的全局搜索，从种群多样性而言，此时种群的多样性匮乏，各个粒子之间的差别很小，无法促使粒子群发展变化[4]。

因此，我将“分裂”的思想引入到了 PSO 算法中，用以保持种群的多样性，防止算法过早产生收敛。

假设有 N 个粒子在 D 维空间进行搜索，则粒子 i 的速度和位置可以采用如下形式表示[5]：

$$v_i = \left(v_{i1}, v_{i2}, \cdots, v_{id}\right)$$
$$x_i = \left(x_{i1}, x_{i2}, \cdots, x_{id}\right)$$

其中，$1 \le i \le N$，$1 \le d \le D$。

算法步骤如下：

1) 初始化每个粒子的速度 v_i 和位置 x_i。当前位置 x_i 则为该粒子目前为止本身最优位置 p_i。根据适应度函数可以得到当前的全局最优位置 p_g。

2) 判断 p_i 和 p_g 之间的欧式距离 d_1 和适应度函数差值与阈值之间的关系。

当二者之间距离过大且适应度函数的差值很小时，证明在距离当前全局最优位置很远的地方也有一个十分优秀的解存在。为了避免陷入局部最优和保持种群的多样性，此时可以将粒子“分裂”为二，一个朝着全局最优位置的方向运动，另一个朝着粒子本身的最优位置的方向运动。

λ_1 是限定 p_i 与 p_g 之间欧式距离 d_1 的阈值。λ_2 是限定适应度函数之间差值的阈值。

在标准 PSO 算法中的速度更新公式如下：

$$v_{id}^{t+1} = \omega v_{id}^{t} + c_1 r_1 \left(p_{id}^{t} - x_{id}^{t}\right) + c_2 r_2 \left(p_{gd}^{t} - x_{id}^{t}\right) \quad (1)$$

其中 $1 \le d \le D$，$1 \le I \le N$，p_{id}^{t} 是 t 时刻粒子本身的最优位置的第 d 维变量，p_{gd}^{t} 是 t 时刻粒子群全局最优位置的第 d 维变量，c_1 和 c_2 是学习因子，r_1 和 r_2 是(0,1)区间上的随机数。如果 $c_1 = 0$，则粒子只“向社会学习”，收敛速度可能很快但容易陷入局部最优，导致“早熟”的问题。如果 $c_2 = 0$，则粒子只“向本身学习”，粒子之间将没有信息交流，导致得到解的几率很小[4]。

当 $s > \lambda_1$ 且适应度函数的差值小于 λ_2 时，该粒子“分裂”为两个粒子。分别令 c_1 和 c_2 等于 0 来采用不同的速度更新公式，使其分别向全局最优位置和本身最优位置的方向运动。

$$v_{id}^{t+1} = \omega v_{id}^{t} + c_2 r_2 \left(p_{gd}^{t} - x_{id}^{t}\right) \quad (2)$$

$$v_{id}^{t+1} = \omega v_{id}^{t} + c_1 r_1 \left(p_{id}^{t} - x_{id}^{t}\right) \quad (3)$$

其它情况时，则采取标准 PSO 算法中的更新公式(1)进行更新。

3) 更新粒子的位置：

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad (4)$$

4) 判断是否符合结束条件。如果适应度函数的值小于允许误差或达到最大迭代次数则停止迭代，输出最优解。否则，重复步骤 2)和 3)。

3. 基于改进 PSO 算法的云任务调度策略

云环境下的任务调度是以任务为出发点，其实质是将相互独立的任务映射到多个可用的资源上，使任务能够高效率高质量的完成。

目前，大部分云计算平台都采用 Google 提出的 Map/Reduce 编程模型，用于大规模数据集的并行计算。通过 Map 和 Reduce 两个阶段，将较大的任务分割成为多个较小的子任务，然后分配给多个资源并行执行，并得到最终的运行结果。在 Map/Reduce 编程模型下，如何对数量众多的子任务进行调度是个复杂的问题[6]。PSO 算法是一种群智能算法[7]，能够很好的解决云环境下的任务调度这一复杂的问题。

3.1. 云中任务和资源的描述形式

用户提交的任务 T 划分为 m 个子任务 t_i，$1 \le i \le m$。则任务 T 可表示为 $T=\{t_1,t_2,\cdots,t_m\}$。每一个子任务都应该有对资源的需求条件，则 $t_i=\{t_{icpu},t_{imen},t_{ina},t_{iio},t_{inet},P_i\}$，分别表示其对资源节点 cpu、内存、网卡、硬盘和带宽的要求，p_i 为该子任务所产生的含有 n 个粒子的粒子群，表示为 $p_i=\{p_{i1},p_{i2},\cdots,p_{ij}\}$，$1 \le j \le n$，$p_{ij}=(thope_i,x_{ij},v_{ij},p_{ij-i})$，$p_{ij}$ 表示第 i 个子任务产生的第 j 个粒子。p_{ij-i} 是该粒子的本身最优位置，x_{ij} 和 v_{ij} 表示第 i 个子任务产生的第 j 个粒子的位置和速度：

$$v_{ij}=(v_{ij-1},v_{ij-2},\cdots,v_{ij-d})$$

$$x_{ij}=(x_{ij-1},x_{ij-2},\cdots,x_{ij-d})$$

云中资源节点集合为 $S=\{s_1,s_2,\cdots,s_q\}$，q 为资源节点个数。每个资源节点表示为 $s_i=(s_{icpu},s_{imem},s_{ina},s_{iio},s_{inet},s_{ilocation})$，其中 $i=1,2,\cdots,q$，分别用来描述该资源节点的 cpu 处理能力、内存、网卡、硬盘和带宽属性。资源 s_i 在 D 维空间中的位置表示为：$s_{ilocation}=(s_{i1},s_{i2},\cdots,s_{id})$。

云环境中粒子和资源之间的距离表示为 d_2：

$$d_2=\sqrt{\sum_{j=1}^{d}(x_{ij}-s_{ij})^2} \tag{5}$$

3.2. 云中任务调度问题的适应度函数

根据子任务对资源各个属性的需求情况，可以得到子任务 t_i 对资源的综合期望：

$$thope_i=\sqrt{\frac{at_{icpu}^2+bt_{imem}^2+ct_{ina}^2+dt_{iio}^2+et_{inet}^2}{a+b+c+d+e}} \tag{6}$$

其中，a、b、c、d、e 为对应的权值，满足条件 $a+b+c+d+e=1$。

云中每个资源 s_i 的资源综合性能可以表示为：

$$sproperty_i=\sqrt{\frac{\alpha s_{icpu}^2+\beta s_{imem}^2+\gamma s_{ina}^2+\chi s_{iio}^2+\delta s_{inet}^2}{\alpha+\beta+\gamma+\chi+\delta}} \tag{7}$$

其中，α、β、γ、χ、δ 分别为对应权值，满足条件 $\alpha+\beta+\gamma+\chi+\delta=1$。

在云环境下任务调度这一问题中，适应度函数应该把任务与资源之间的匹配程度作为衡量标准，所以函数如下：

$$F=\min_{d_2<\lambda_3}\{|thope_i-sproperty_i|\} \tag{8}$$

其中，λ_3 是限制粒子与资源之间距离的阈值。

3.3. 云中任务调度的具体步骤

在云计算环境中，在采用基于“分裂”思想的 PSO 算法进行任务调度时，具体步骤可以划分为如下五步：

1) 接收用户提交的任务，并把每个任务划分为 m 个子任务，为每个子任务生成粒子群。

2) 初始化粒子群中每个粒子的速度 v_{ij} 和位置 x_{ij}。在当前时刻，第 i 个子任务的第 j 个粒子的本身最优位置 $p_{ij-i}=x_{ij}$。利用上述适应度函数(8)，可以得到当前第 i 个子任务产生的所有粒子的全局最优位置 p_{i-g}。

3) 判断 p_{ij-i} 和 p_{i-g} 之间的欧式距离 d_1 和适应度函数差值与阈值 λ_1、λ_2 之间的关系。

当 $d_1>\lambda_1$ 且 F 之差 $<\lambda_2$ 时，开始“分裂”为两个粒子，分别采用新的速度更行公式：

$$v_{ij-d}^{t+1}=\omega v_{ij-d}^{t}+c_2r_2\left(p_{i-g-d}^{t}-x_{ij-d}^{t}\right) \tag{9}$$

$$v_{ij-d}^{t+1}=\omega v_{ij-d}^{t}+c_1r_1\left(p_{ij-i-d}^{t}-x_{ij-d}^{t}\right) \tag{10}$$

其它情况时，则采取标准 PSO 算法中的速度更新公式：

$$\begin{aligned}v_{ij-d}^{t+1}=\omega v_{ij-d}^{t}&+c_1r_1\left(p_{ij-i-d}^{t}-x_{ij-d}^{t}\right)\\&+c_2r_2\left(p_{i-g-d}^{t}-x_{ij-d}^{t}\right)\end{aligned} \tag{11}$$

4) 更新每个粒子的位置，公式如下：

$$x_{ij-d}^{t+1}=x_{ij-d}^{t}+v_{ij-d}^{t+1} \tag{12}$$

5) 判断是否满足结束条件。

如果 $d_2=0$ 且 $F<\lambda_4$（λ_4 是适应度函数的最大允许误差），表示粒子到达了一个合适资源的位置，即已经找到合适的资源，则把任务分配到该资源节点执行，结束搜寻。

如果迭代次数超过最大迭代次数 λ_5，则也结束搜寻，重新初始化粒子群，重复步骤 2)、3)和 4)。

如果上述两个条件都不满足，则继续迭代，重复步骤 2)、3)和 4)。

4. 仿真实验结果

为了验证改进后的基于“分裂”思想的 PSO 算法，采用了 CloudSim[8]做模拟，并将其与标准 PSO 算法、遗传算法和蚁群算法进行比较。

采用 30 个资源节点来模拟实验，cpu 度量为频率系数，内存度量为容量系数，硬盘为 I/O 读取系数，网卡为通信速度系数，带宽为带宽系数(如表 1 所示)。

权值 a、b、c、d、e、α、β、γ、χ 和 σ 均取 0.2。经过反复多次试验，最终选取各个阈值的值如下：

Table 1. the description of resource node attribute parameter
表 1. 资源节点的属性参数描述

资源节点编号	cpu(G)	Mem(G)	na(m/s)	i/o(m/s)	Net(m/s)
1	2.4	1	100	300	100
2	1.6	1	100	300	100
3	1.6	1	100	300	100
4	2.2	1	100	300	100
5	4.2	1	100	300	100
6	4.2	1	100	300	100
7	1.6	0.5	100	300	100
8	1.8	0.5	100	300	100
9	1.8	0.5	100	300	100
10	4.2	1	100	300	100
11	7.2	3.6	100	300	100
12	7.2	3.6	100	300	100
13	4.4	1	100	300	100
14	4.4	1	100	300	100
15	4.4	0.5	1000	300	100
16	4.6	1	1000	150	100
17	4.6	1	1000	150	100
18	4.6	1	1000	150	100
19	1.6	0.5	100	150	100
20	4.2	1	1000	150	100
21	4	0.25	100	150	100
22	3	0.25	100	150	100
23	3	0.25	100	150	100
24	2.8	0.25	100	150	100
25	4.6	1	100	150	100
26	3	0.5	100	150	100
27	3	0.5	100	150	100
28	2.8	1	100	150	100
29	3	0.5	100	150	100
30	3	0.5	100	150	100

$$\lambda_1 = 100,\ \lambda_3 = \lambda_1 / 4.0 = 25.0,\ \lambda_4 = 0.2,$$
$$\lambda_2 = 2\lambda_4 = 0.4,\ \lambda_5 = 30$$

提交任务大小为 200 M，划分为 10 个子任务进行处理，为每个子任务生成 10 个粒子，随机初始化每个粒子的位置和速度。

4.1. 与标准 PSO 算法在解决早熟问题上的比较

如图 1 所示，标准 PSO 算法在迭代到第 14 次时，适应度函数的值小于允许误差而停止迭代，最优资源的适应度函数值为 0.17。改进后的基于“分裂”思想的 PSO 算法在迭代 13 次时找到了最优资源，适应度函数值为 0.12，可见该资源优于标准 PSO 算法找到的资源。为了验证改进后的基于“分裂”思想的 PSO 算法没有造成“早熟”，对其又进行了 10 次迭代，发现适应度函数的值没有发生变化，这证明改进后的算法在解决“早熟”问题上十分有效，很好的保持了种群的多样性。

4.2. 与标准 PSO 算法、遗传算法和蚁群算法在任务完成时间方面的比较

提交五个任务大小分别为：50 M、100 M、150 M、200 M 和 250 M。分别采用不同的算法，记录 5 个任务的完成时间。

如图 2 所示，改进后的基于“分裂”思想的 PSO 算法与其他三种算法相比在完成时间上有明显的减少。由于改进后的算法在特定情况下将粒子一分为二，从而增加了粒子的数目，粒子之间的共享信息也有所增加，这将有利于提高搜寻的速度，搜索时间也有所减少。

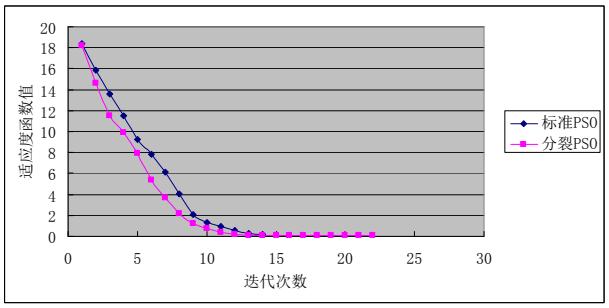

Figure 1. Comparison of the “Separatist” PSO algorithm and standard PSO algorithm on the problem of “premature”
图 1. “分裂”PSO 算法与标准 PSO 算法在“早熟”问题上的比较

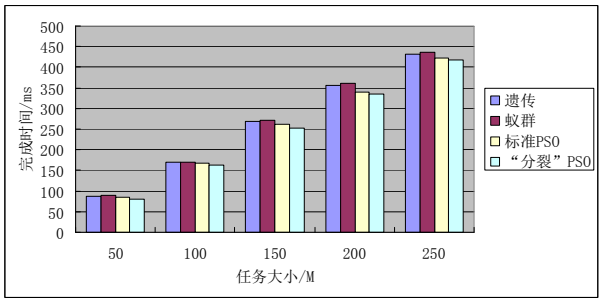

Figure 2. Comparison of the “Separatist” PSO algorithm, the standard PSO algorithm, genetic algorithm and ant colony algorithm in task completion time

图 2. “分裂”PSO 算法与标准 PSO 算法、遗传算法和蚁群算法在任务完成时间方面的比较

5. 结论

通过实验可以验证，基于“分裂”思想的 PSO 算法很好的解决了标准 PSO 算法容易“早熟”的问题，有效的保持了种群的多样性。这将有利于提高任务和资源之间的匹配度和缩小完成任务的时间。与其他仿生算法相比，基于“分裂”思想的 PSO 算法能够比遗传算法和蚁群算法更快的完成任务与资源节点之间的映射。

6. 致谢

本文获得了“辽宁省教育厅科学研究一般项目(L2011004)”基金项目的支持，在此深表感谢。

参考文献 (References)

[1] 郭本俊, 王鹏, 陈高云等. 基于 MPI 的云计算模型[J]. 计算机工程, 2009, 35(24): 84-86.

[2] J. Kenney, R. C. Eberhart. Particle swarm optimization. Institute of Electrical and Electronics Engineers, 1995, 11: 1942-1948.

[3] 李丽, 牛奔. 粒子群优化算法[M]. 北京: 冶金工业出版社, 2009: 10-26.

[4] 毛恒. 粒子群优化算法的改进及应用研究[D]. 华侨大学, 2007.

[5] 韦杏琼, 周永权, 黄华娟等. 云自适应粒子群算法[J]. 计算机工程与应用, 2009, 45(1): 48-50.

[6] S. Chemawat, H. Gobioff and T. Shun. The google file system. ACMSIGOPS Operating Systems Review, 2003, 37(5): 29-43.

[7] 高尚, 杨静宇. 群智能算法及其应用[M]. 北京: 中国水利水电出版社, 2006: 1-2.

[8] R. N. Calheriros, R. Ranjan and R. Buyya. CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services. Melbourne: Grid Computing and Distributed Systems Laboratory, The University of Melbourne, 2009.

