第 39 卷 第 5 期
Vol.39 No.5
计 算 机 工 程
Computer Engineering
2013 年 5 月
May 2013

·人工智能及识别技术· 文章编号：1000—3428(2013)05—0183—04 文献标识码：A 中图分类号：TP393

云计算环境下基于改进粒子群的任务调度算法

封良良[1]，张 陶[1]，贾振红[1]，夏晓燕[2]，覃锡忠[1]

(1. 新疆大学信息科学与工程学院，乌鲁木齐 830046；2. 中国移动通信集团新疆有限公司，乌鲁木齐 830063)

摘 要：现有云计算任务调度算法为追求最短完成时间不能很好地兼顾成本。为此，提出一种基于改进粒子群的任务调度算法。采用间接编码方式对每个子任务占用的资源进行编码，给出解码方式，定义考虑时间和成本的适应度函数，确立粒子位置和速度的更新方法。实验结果表明，在相同的条件设置下，该算法的总任务完成时间和总任务完成成本小于传统粒子群优化算法。

关键词：云计算；任务调度；时间成本；双适应度粒子群优化；粒子群优化算法

Task Schedule Algorithm Based on Improved Particle Swarm Under Cloud Computing Environment

FENG Liang-liang[1], ZHANG Tao[1], JIA Zhen-hong[1], XIA Xiao-yan[2], QIN Xi-zhong[1]

(1. School of Information Science and Engineering, Xinjiang University, Urumqi 830046, China;
2. Subsidiary Company of China Mobile in Xinjiang, Urumqi 830063, China)

【Abstract】 Existing task schedule algorithms for cloud computing are not well to take into account the cost of all the tasks for the pursuit of the shortest completion time. To solve this problem, a task schedule algorithm based on improved particle swarm is proposed in this paper, which uses indirect encoding to encode the resources of each subtask takes, gives decoding way, considers the fitness function about the time and the cost, and establishes the particle position and velocity updating method. Experimental results show that the general assignment finished time and cost of this algorithm are lower than the traditional Particle Swarm Optimization(PSO) algorithm.

【Key words】 cloud computing; task schedule; time cost; Double-fitness Particle Swarm Optimization(DFPSO); Particle Swarm Optimization(PSO) algorithm

DOI: 10.3969/j.issn.1000-3428.2013.05.040

1 概述

近几年，云计算[1-2]成为了人们讨论的热点。它是网格计算、并行计算[3]的发展，并在商业领域[4-5]得以实现，其最基本的思想是通过网络将庞大的计算处理任务分拆成多个较小子任务，再交由多部服务器所组成的庞大系统，经搜寻、计算分析之后将计算结果回传给用户。

由于云计算所面对的计算任务数量十分庞大，任务调度和资源分配问题是决定云计算效率的重点与难点。目前，对云计算中任务调度和资源分配的相关研究不多，现有的资源调度算法的研究重点还放在缩短任务完成时间上，如遗传算法[6]、Min-Min 算法[7]、Sufferage 算法[8]等。然而，它们没有很好地兼顾调度执行时间最小与成本最小问题，因为在云计算模型中，任务执行所需要的成本也是一个不可忽略的因素，不同计算能力的资源，其使用成本也不同。计算能力弱一些的资源，其使用成本较低；而计算能力强的资源，其使用成本较高。文献[9]提出一种基于双适应度的遗传算法对任务进行调度，考虑总任务完成时间和任务平均完成时间 2 个因素。然而，没有考虑总任务完成成本，本文则考虑总任务完成时间和总任务完成成本这 2 个因素，在传统粒子群优化(Particle Swarm Optimization, PSO)算法的基础上，本文选取双适应度的原则[10]，提出一种基于改进粒子群的任务调度算法，以期在实现最小执行时间的同时兼顾成本最小。

2 云计算中的任务调度问题

在云计算环境中，一个大规模计算任务的处理必须进行分布式并行处理。首先要将一个逻辑上完整的大任务分

基金项目：中国移动新疆分公司研究发展基金资助项目(xjm2011-1)

作者简介：封良良(1986－)，男，硕士研究生，主研方向：云计算，人工智能；张 陶，硕士研究生；贾振红，教授、博士；夏晓燕，工程师、硕士；覃锡忠，副教授、硕士

收稿日期：2012-06-04 **修回日期**：2012-08-07 **E-mail**：liangzai0108@126.com

解成若干个子任务，系统根据任务的信息采用适当的策略把不同的任务分配到不同资源节点上去运行。当所有子任务处理结束，则完成整个大任务的一次处理，将处理结果传给用户。目前，云计算环境中大部分采用 Google 提出的 Map/Reduce[11]编程模型。一个并行处理任务由多个 Map 任务和多个 Reduce 任务组成，任务的执行分为 Map 阶段和 Reduce 阶段。在 Map 阶段，每个 Map 任务对分配给它的数据进行计算，然后按照 Map 的输出 key 值将结果数据映射到对应的 Reduce 任务中；在 Reduce 阶段，每个 Reduce 任务对接收到的数据做进一步聚集处理，从而得到输出结果。

由于云计算可扩展性和动态性的特点，任务量和资源的数量是相当庞大的，系统每时每刻都要处理海量的任务，因此在 Map/Reduce 编程模型下，如何对众多的子任务进行调度是个复杂的问题。任务调度策略的好坏直接影响任务的执行时间，进而影响整个云的性能以及用户的使用满意度。目前，一些较为典型的任务调度算法多是以总任务完成时间作为调度目标，而没有太多考虑总任务完成成本，这容易造成总任务完成时间较短而成本较高的情况。因此，本文采用考虑时间和成本约束，对云计算中任务调度和资源分配策略进行改进，以期最大限度地提高云计算效率。

3 本文算法

粒子群算法[12]是一种群智能优化算法，它是一种仿生优化算法，来源于对鸟类和鱼类觅食过程的模拟。传统 PSO 算法在求解云计算任务调度这个大规模的、实时的问题上有着天然的优势。

3.1 问题编码

粒子的编码有多种方式，可以采用直接编码(基于问题直接对粒子的位置速度进行编码，一个粒子对应问题的一个可行解)，也可以采用间接编码方式。本文采用间接编码方式，对每个子任务占用的资源进行编码，编码长度取决于子任务数量，这样一个粒子实际上对应着一个任务分配策略。

设有 *TASK* 个任务，*RESOURCE* 个资源，每个任务又分为若干个子任务，且任务划分的子任务总数 *SUBTASK* 大于资源数 *RESOURCE*，即 *SUBTASK*>*RESOURCE*。

子任务的总数量为：

$$subTaskNum=\sum_{t=1}^{TASK} taskNum(t) \tag{1}$$

其中，*taskNum*(*t*)为第 *t* 个任务所含子任务的个数。

对子任务的编码为：

$$m[i,j]=\sum_{k=1}^{i-1} taskNum(k)+j \tag{2}$$

采用顺序编码法，即按任务顺序进行编码。第 *i* 个 *task* 中的第 *j* 个 *subTask* 的序号是 *m*[*i*,*j*]。

当 *TASK*=3，*RESOURCE*=3，*SUBTASK*=10 时，粒子(2,3,1,2,3,3,2,1,2,1)即为一个可行的调度策略。粒子编码如表 1 所示，其中，任务、子任务对(2,5)表示第 2 个任务中的第 3 个子任务序号是 5；子任务、资源对(1,2)表示把子任务 1 分配到资源 2 上执行。

表 1 粒子编码示例

任务序号	子任务序号	资源号
1	1	2
	2	3
2	3	1
	4	2
	5	3
3	6	3
	7	2
	8	1
	9	2
	10	1

对粒子解码得到资源上的 *subTask* 分布情况。粒子解码示例如表 2 所示，其中，资源 1 上执行的子任务是{3,8,10}；资源 2 上执行的子任务是{1,4,7,9}；资源 3 上执行的子任务是{2,5,6}。

表 2 粒子解码示例

资源序号	子任务号
1	3
	8
	10
2	1
	4
	7
	9
3	2
	5
	6

本文使用 ***ETC*** 矩阵[13]记录任务的执行时间，*ETC*[*i*,*j*]表示子任务 *i* 在第 *j* 个资源上执行的时间。用 *RCU* 数组表示各个计算资源单位时间任务运行的成本，*RCU*[*r*]表示第 *r* 个计算资源单位时间任务运行的成本。

完成所有任务的总时间为：

$$SFT=\max_{r=1}^{RESOURCE}\sum_{i=1}^{n} resource(r,i) \tag{3}$$

其中，*resource*(*r*,*i*)是第 *r* 个资源执行该资源上第 *i* 个子任务所用的时间；*n* 为分配到此资源上的子任务个数。

第 *t* 个任务完成的时间为：

$$taskTime(t)=\max_{i=1}^{taskNum(t)}\sum_{j=1}^{k} res(j,i) \tag{4}$$

其中，*res*(*j*,*i*)是子任务 *i* 分配的资源上执行该资源第 *j* 个子任务所用的时间；*k* 为第 *t* 个任务的第 *i* 个子任务在被分到的资源中的位置。

第 *r* 个资源执行该资源上的所有子任务所用的时间为：

$$sumTime(r)=\sum_{i=1}^{n} resource(r,i) \tag{5}$$

完成所有任务的总成本为：

$$sumCost(r)=\sum_{r=1}^{RESOURCE} sumTime(r)\times RCU(r) \tag{6}$$

3.2　粒子群体初始化

种群规模为 S，子任务总数为 $SUBTASK$，资源的数量为 $RESOURCE$，则初始化描述为：系统随机产生 S 个粒子，第 i 个粒子位置由向量 $\boldsymbol{x}_i$ 表示，定义 $\boldsymbol{x}_i$ 为 $\boldsymbol{x}_i=\{x_{i1}, x_{i2},\cdots,x_{in}\},(1\leqslant n\leqslant SUBTASK,1\leqslant i\leqslant S)$。其中，$x_{ij}$ 表示任务 j 分配到节点 $\boldsymbol{x}_i$ 上$(1\leqslant x_{ij}\leqslant RESOURCE)$。速度由 $\boldsymbol{v}_i$ 位向量表示 $\boldsymbol{v}_i=\{v_{i1},v_{i2},\cdots,v_{in}\},(1\leqslant n\leqslant SUBTASK,1\leqslant i\leqslant S)$。其中，$-RESOUR-CE\leqslant v_{ij}\leqslant RESOURCE$。初始化粒子位置为 [1,$RESOURCE$] 之间的整数，粒子的速度随机取 [−($RESOURCE$−1), ($RESOURCE$−1)]之间的整数。

3.3　适应度函数

粒子群算法的适应度函数选取至关重要，直接影响到粒子群算法的收敛速度与最优解的查找。任务调度较为典型的多是以任务总完成时间来作为调度目标，传统 PSO 算法用任务总完成时间来定义适应度函数，而本文算法在考虑任务总完成时间的同时，考虑总任务完成的成本，即定义 2 个适应度函数。

定义时间的适应度函数为：

$$F_t(i)=\frac{1}{SFT_i},1\leqslant i\leqslant s \tag{7}$$

其中，SFT_i 为第 i 个粒子的任务总完成时间，由式(3)计算得出。

定义成本的适应度函数为：

$$F_c(i)=\frac{1}{sumCost(i)},1\leqslant i\leqslant s \tag{8}$$

本文采用类似遗传算法中选择的思想，优先选择适应度高的粒子。任务总完成时间和任务完成总成本越小的粒子，适应度值越大，越容易被选择。种群中既有总任务完成时间较短的粒子，又有任务完成总成本较小的粒子，为进化出下一代较优秀的粒子提供了优良的基础。

3.4　粒子位置与速度的更新

在每一次迭代中，粒子依据自身的历史最优位置和整个群体的最优位置来更新速度和位置，只有当粒子的当前位置与所经历的最好位置相比具有更好的适应值时，其粒子所经历的最好位置才会唯一地被该粒子当前的位置所替代[14]。第 i 个粒子经历过的最好位置(有最好适应度)记为 $pb_i=(pb_{i1}, pb_{i2},\cdots, pb_{in})$，在整个群体中，所有粒子经历过的最好位置记为 $gb=(gb_1, gb_2,\cdots, gb_n)$。$s$ 代表群体的大小。分别用式(7)、式(8)得出的 2 个适应度计算 pb、gb。

$$pb_i(t+1)=\begin{cases} pb_i(t) & \text{if } f_t(x_i(t+1))<f_t(pb_i(t)) \\ pb_i(t) & \text{if } f_t(x_i(t+1))=f_t(pb_i(t))\cap \\ & \quad f_c(x_i(t+1))\leqslant f_c(pb_i(t)) \\ x_i(t+1) & \text{if } f_t(x_i(t+1))=f_t(pb_i(t))\cap \\ & \quad f_c(x_i(t+1))>f_c(pb_i(t)) \\ x_i(t+1) & \text{if } f_t(x_i(t+1))>f_t(pb_i(t)) \end{cases} \tag{9}$$

$$f_tMax(t)=getMax\{f_t(pb_1(t)),f_t(pb_2(t)),\cdots,f_t(pb_s(t))\} \tag{10}$$

其中，$f_tMax(t)\in\{pb_1,pb_2,\cdots,pb_s\}$。

$$gb(t)=\begin{cases} f_tMax(t) & \text{if } length(f_tMax(t))=1 \\ \max\begin{Bmatrix} f_c(f_tMax(t)[1]),f_c(f_tMax(t)[2]),\cdots, \\ f_c(f_tMax(t)[n]) \end{Bmatrix} & \text{if } length(f_tMax(t))>1 \end{cases} \tag{11}$$

其中，式(11)表示当有多个粒子具有相同的时间适应度值时，选择具有最佳成本适应度值的粒子作为全局最优粒子的判断规则。

每一代粒子根据下式更新自己的速度和位置：

$$v_i(t+1)=w\times v_i(t)+c_1\times Rand()\times(pb_i(t)-x_i(t))+ c_2\times Rand()\times(gb(t)-x_i(t)) \tag{12}$$

其中，第 1 部分可理解为粒子先前的速度或惯性；第 2 部分可理解为粒子的“认知”行为，表示粒子本身的思考能力；第 3 部分可理解为粒子的“社会”行为，表示粒子之间的信息共享与相互合作。即首先利用粒子自身的最佳飞行位置 pb[15]作用于当前位置，然后根据群体最佳位置 gb 对当前粒子位置进行调整。

$$x_i(t+1)=x_i(t)+v_i(t) \tag{13}$$

其中，t 表示迭代次数；w 为惯性权重，使粒子保持运动惯性，防止算法的早熟收敛；c_1 和 c_2 是学习因子，$Rand$ 为[0,1]之间的随机数。在迭代过程中，粒子的速度和位置都限制在特定的范围内，同时 pb 和 gb 不断更新，最后输出的 gb 就是全局最优解。

4　实验结果与分析

4.1　实验仿真环境及设置

为了评价和分析本文算法的性能，实验采用 Matlab 产生 ***ETC*** 矩阵和 RCU 数组，首先用 CloudSim[16]对传统 PSO 算法和本文算法进行了云环境下的仿真实验，继而在相同的仿真平台下对这 2 种算法进行对比。实验测试运行 100 次，取 100 次实验的平均结果作为作图的数据。

通过多次实验并参考文献[17]的参数调整策略，2 种算法参数按表 3 设置，可以在较短的时间内取得优质解。

表 3　参数设置

名称	取值
种群规模 S	150
计算资源数 $RESOURCE$	50
任务数 $TASK$	50
惯性因子 w	0.9
学习因子 c_1	2
学习因子 c_2	2
最大迭代次数 t_{max}	200

由表 3 可知，每个任务划分成的子任务个数范围为[30,70]，各个子任务的预计完成时间在[1,30]区间上随机产生，单位为秒。算法终止条件为：(1)达到最大迭代次数 t_{max}；(2)连续 60 次总任务完成时间与总任务完成成本都没有变化。

4.2 性能分析

任务数为 50 时的总任务完成时间和总任务完成成本如图 1、图 2 所示。

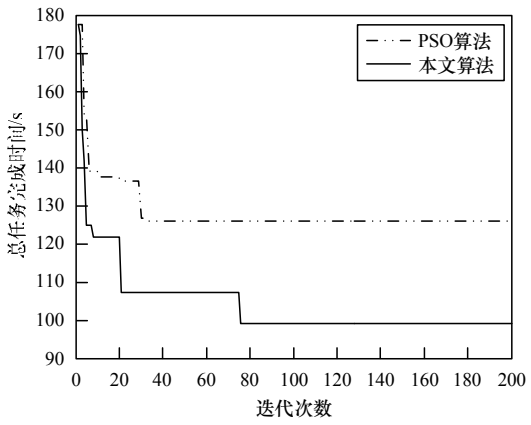

图 1 任务数为 50 时的总任务完成时间

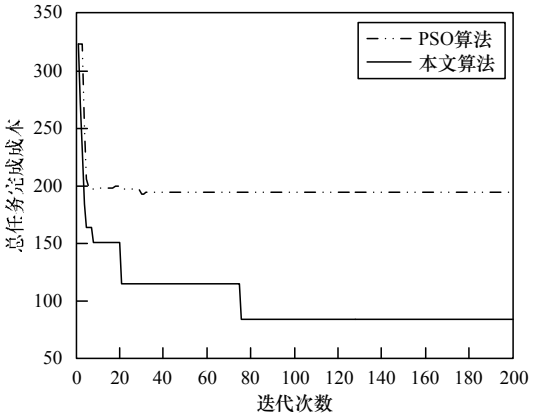

图 2 任务数为 50 时的总任务完成成本

由图 1 和图 2 可知，PSO 算法和本文算法的前期迭代过程基本一致，得出的任务总完成时间和任务完成所需成本相差不大。随着粒子迭代更新次数的增大，2 种算法所得到的任务总完成时间和任务完成所需成本都在不断地减小，但本文算法所得调度结果在任务总时间和任务完成所需成本上要小于传统 PSO 算法。

从以上实验结果反映出，传统 PSO 算法因为只重视总任务的完成时间，造成了一些潜在优良的粒子丢失。虽然提高了自己的收敛速度，但该算法在运行过程中由于无法有效跳出局部最优的搜索状态而过早地收敛到局部最优的任务调度结果上，得到的任务调度结果不论是任务总完成时间还是任务完成所需成本都较大。本文在传统 PSO 算法的基础上增加一个适应度，该算法由于同时考虑任务总完成时间和任务完成所需成本，尽管在收敛速度上比 PSO 算法有所下降，但该算法任务的调度结果不但任务总完成时间短，任务完成所需成本也较小。

5 结束语

任务调度算法通常以总任务完成时间作为标准对任务进行调度，而没有将任务完成所需成本考虑在内，本文在传统粒子群优化算法的基础上，针对云计算的编程模型，提出了一种改进的任务调度算法。在进行任务调度时不仅考虑总任务完成时间，同时也考虑到了任务完成所需的总成本。实验结果表明，该算法可以在云计算环境下实现较为理想的任务调度结果。今后将研究云计算中的数据分布等因素对任务调度结果的影响。

参考文献

[1] Armbrust M, Fox A, Griffith R, et al. Above the Clouds: A Berkeley View of Cloud Computing[EB/OL]. (2009-11-21). http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf.

[2] Foster I, Zhao Yong, Raicu I, et al. Cloud Computing and Grid Computing 360-degree Compared[C]//Proc. of Grid Computing Environments Workshop. Washington D. C., USA: IEEE Computer Society, 2008.

[3] Chien A, Calder B, Elbert S, et al. Entropia: Architecture and Performance of an Enterprise Desktop Grid System[J]. Journal of Parallel and Distributed Computing, 2003, 63(5): 597-610.

[4] Rochwerger B, Breitgand D, Levy E, et al. The Reservoir Model and Architecture for Open Federated Cloud Computing[J]. IBM Journal of Research and Development, 2009, 53(4): 1-17.

[5] Daniel N, Wolski R, Grzegorczyk C, et al. The Eucalyptus Open-source Cloud-computing System[C]//Proc. of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. Washington D. C., USA: IEEE Computer Society, 2009.

[6] 罗 红, 慕德俊, 邓智群, 等. 网格计算中任务调度研究综述[J]. 计算机应用研究, 2005, 22(5): 16-19.

[7] Buyya R. Economic-based Distributed Resource Management and Scheduling for Grid Computing[D]. Melbourne, Australia: Monash University, 2002.

[8] 林剑柠, 吴慧中. 基于遗传算法的网格资源调度算法[J]. 计算机研究与发展, 2004, 41(12): 2195-2199.

[9] 李建锋, 彭 舰. 云计算环境下基于改进遗传算法的任务调度算法[J]. 计算机应用, 2011, 31(1): 184-187.

[10] 张粒子, 陈之栩, 舒 隽. 基于微粒群优化算法的阻塞管理[J]. 中国电机工程学报, 2005, 25(22): 73-77.

(下转第 191 页)