文章编号:1001 - 9081(2011)01 - 0184 - 03
doi:10.3724/SP. J. 1087. 2011. 00184

云计算环境下基于改进遗传算法的任务调度算法

李建锋,彭　舰
(四川大学 计算机学院,成都 610065)
(jianpeng@ scu. edu. cn)

摘　要:在云计算中面对的用户群是庞大的,要处理的任务量与数据量也是十分巨大的。如何对任务进行高效的调度成为云计算中所要解决的重要问题。针对云计算的编程模型框架,提出了一种具有双适应度的遗传算法(DFGA),通过此算法不但能找到总任务完成时间较短的调度结果,而且此调度结果的任务平均完成时间也较短。通过仿真实验将此算法与自适应遗传算法(AGA)进行比较,实验结果表明,此算法优于自适应遗传算法,是一种云计算环境下有效的任务调度算法。

关键词:云计算;遗传算法;双适应度;任务调度
中图分类号: TP393　　**文献标志码**:A

Task scheduling algorithm based on improved genetic algorithm in cloud computing environment

LI Jian-feng, PENG Jian
(*College of Computer Science, Sichuan University, Chengdu Sichuan* 610065, *China*)

Abstract: The number of users is huge in cloud computing, and the number of tasks and the amount of data are also huge. How to schedule tasks efficiently is an important issue to be resolved in cloud computing environment. A Double-Fitness Genetic Algorithm (DFGA) was brought up for the programming framework of cloud computing. Through this algorithm, the better task scheduling not only shortens total-task-completion time and also has shorter average-completion time. There is a contrast between DFGA and Adaptive Genetic Algorithm (AGA) through simulation experiment, and the result is: the DFGA is better, it is an efficient task scheduling algorithm in cloud computing environment.

Key words: cloud computing; Genetic Algorithm (GA); double-fitness; task scheduling

0　引言

近几年云计算[1-2]成为了人们讨论的热点。目前 IBM、Google、Amazon、Microsoft 等纷纷涉足云计算,提供了众多基于云计算的服务,如 Gmail、Google Earth、Google Analytics、Google 搜索、Google 文档[3];Amazon 的弹性云计算(EC2)服务和存储服务(S3);Microsoft 的 Windows Live Web 应用套件及 Hotmail 等[4]。

云计算是并行计算、网格计算[5-6]的发展,是分布式计算的一种,其最基本的思想是透过网络将庞大的计算处理程序自动分拆成无数个较小的子程序,再交由多部服务器所组成的庞大系统,经搜寻、计算分析之后将处理结果回传给用户,提供这些资源的网络被称为“云”。云计算所提供的服务面向的用户群是庞大的,因此“云”中的任务数量是巨大的,系统每时每刻都要处理海量的任务,所以任务调度[7]是云计算中的重点与难点。本文对如何充分利用“云”中的资源使其中的任务进行高效合理的调度进行了研究,提出了一种基于双适应度遗传算法(Double-Fitness Genetic Algorithm, DFGA)的任务调度算法,并通过了仿真实验,验证了其良好的性能。

1　云计算中的编程模型

目前的云计算环境中大部分采用 Google 提出的 Map/Reduce 的编程模式[8],大部分信息技术厂商提出的“云”计划中采用的编程模型,都是采用基于 Map/Reduce 的思想开发的编程工具,它特别适用于产生和处理大规模的数据集。其执行过程如图 1 所示。

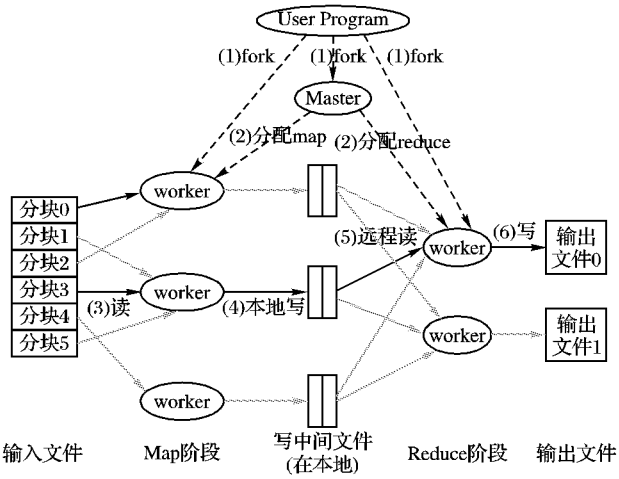

图 1　Map/Reduce 的具体执行过程

从图 1 可以看出,Map/Reduce 有 6 个过程,可分为两个主要阶段。

Map 阶段　把一个较大的任务通过 Map/Reduce 函数分割为 M 个较小的子任务,然后配给多个 worker(被分配为执行 Map 操作的 worker)并行执行,输出处理后的中间文件;

收稿日期:2010 - 07 - 15;**修回日期**:2010 - 09 - 06。　**基金项目**:四川省科技支撑计划项目(06KJT-013;2009GZ0153)。

作者简介:李建锋(1987 -),男,河南项城人,硕士研究生,主要研究方向:网格计算、云计算;　彭舰(1970 -),男,四川成都人,教授,博士,主要研究方向:分布式系统、移动计算。

Reduce 阶段　将 Map 阶段处理后的结果进行汇总分析处理,输出最后的结果:R 个输出文件(R 为 Reduce 任务的个数)。

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。在云计算中,同时向许多用户提供服务,要考虑到每个用户的响应时间,不能让有些用户等待的时间太长。同时也要考虑用户的综合满意度,这就要用任务的平均完成时间来衡量。而目前的一些任务调度算法只关注了总任务的完成时间,即用户的最大等待时间,没有太多地关注任务的平均完成时间,这样会造成许多潜在的优良基因的丢失,因为当任务的平均完成时间较小时,使找到更小的任务总完成时间成为了可能;并且在云计算中又把一个用户程序分为许多子任务来执行,这样就更容易造成总的任务完成时间较为理想,而任务的平均完成时间较大的情况。因此本文提出了 DFGA 对云计算中的任务调度策略进行改进,通过对任务的优化调度来最大限度地提高云计算环境的效率。

2　采用 DFGA 进行"云"中任务调度

遗传算法(Genetic Algorithm,GA)是 Holland 于 1975 年受生物进化论的启发而提出的,并行性和全局解空间搜索是 GA 的两个最显著的特点[9]。针对云计算环境下的任务调度问题,用遗传算法进行任务调度能得到较好的效果。参照 Map/Reduce 模型[8],并且为了能得到总任务执行时间和任务平均执行时间都较短的任务调度结果,本文对遗传算法作了一些改进,增加了一个适应度,用两个适应度来选择种群,即双适应度遗传算法(DFGA)。

2.1　染色体编码与解码

染色体编码有很多种方式,可以采用直接编码,即直接对任务的执行状态编码,也可以采用间接编码。本文采用资源——任务的间接编码方式。染色体的长度为子任务的数量,染色体中每个基因的取值为该位置号对应的子任务分配到资源上的资源编号。

设有 $TASK$ 个任务(task),$WORKER$ 个 worker 资源,第 t 个任务被分为的子任务(subTask)的数量为:$taskNum(t)$。则子任务的总数量为 $subTaskNum$:

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

如假设有 3 个大任务(task),3 个 worker 资源,每个 task 又分为若干个 subTask:

如 task1 被分为:$t1.1$,$t1.2$,$t1.3$ 三个 subTask;task2 被分为:$t2.1$,$t2.2$;task3 被分为:$t3.1$,$t3.2$,$t3.3$,$t3.4$,$t3.5$。则总共有 10 个 subTask,再对这些 subTask 进行编号,其中较简单的一种编法是:依次按任务顺序——子任务顺序编码法。即第 i 个 task 中的第 j 个 subTask 的序号为 m:

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

染色体的长度为 10,每个基因的取值为 1 ~ 3,如产生下面的一个染色体:

{2,3,1,2,2,3,2,1,1,1}

则这条染色体代表第 1 个 subTask 在第 2 个 worker 上执行,第 2 个 subTask 在第 3 个 worker 上执行,……,第 10 个 subTask 在第 1 个 worker 上执行。

之后就是对染色体解码,得到 worker 上的 subTask 分布情况。生成多组以资源编号的 subTask 序列。如将上述染色体解码为:

W1:{3,8,9,10}　W2:{1,4,5,7}　W3:{2,6}

通过解码后的序列和 ETC(Expected Time to Compute)矩阵($ETC[i,j]$ 表示第 i 个 subTask 在第 j 个资源上执行完成所要用的时间)可以计算出每个资源执行该资源上的所有子任务所用的时间,则完成所有任务的总时间函数为:

$$F_1(x) = \max_{w=1}^{WORKER} \sum_{i=1}^{n} worker(w,i) \tag{3}$$

其中:$worker(w,i)$ 为第 w 个 worker 执行该 worker 上第 i 个子任务所用的时间,n 为分配到该 worker 上的子任务数量。

同时通过解码后的序列和 ETC 矩阵可以计算出第 t 个 task 的完成时间:

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

其中:k 为 $task(t)$ 的第 i 个子任务在被分到的 worker 中的位置,$W(j,i)$ 为在子任务 i 分配的 worker 上执行此 worker 第 j 个子任务所用的时间。

则任务的平均所用时间函数:

$$F_2(x) = \frac{\sum_{t=1}^{TASK} taskTime(t)}{TASK} \tag{5}$$

2.2　初始种群生成

若种群规模为 $SCALE$,子任务总个数为 M,资源数即 worker 的个数为 $WORKER$,则初始化描述为:由系统随机产生 $SCALE$ 个染色体,染色体的长度为 M,基因的取值范围为[1,$WORKER$],在其中随机取值。

2.3　适应度函数

遗传算法是通过一定的适应度函数来进行下一代的选择进化,从而去寻找问题的最优解。因此适应度的选取是相当重要的,关系到算法的收敛速度与所得解的优劣。

在任务调度中一个重要的目标是:总任务的完成时间短。但任务的平均所用时间也不能忽视,把任务的平均所用时间也考虑后,有利于算法收敛速度的提高,也是找到实际最优解所必需的,即:不但总任务完成时间短,而且任务的平均所用时间也短。因此定义两个适应度函数:

$$f_1(i) = 1/SCD_i(W_j);\ 1 \leqslant i \leqslant SCALE, 1 \leqslant j \leqslant WORKER \tag{6}$$

W_j 为第 i 个个体中的第 j 个 worker,$SCD_i(W_j)$ 为第 i 个个体完成总任务所用的时间。

$$f_2(i) = meantime(i) = \frac{\sum_{t=1}^{TASK} taskTime(t,i)}{TASK};\quad 1 \leqslant i \leqslant SCALE \tag{7}$$

$taskTime(t,i)$ 为在第 i 个个体中完成第 t 个任务所用的时间。

即总任务完成时间和任务平均所用时间越短的个体,适应度值越大,越容易被选择。

2.4　遗传操作

2.4.1　个体选择

选择操作是遗传算法对个体适应性的评价方式,也是实现群体优良基因传播的基本方式。DFGA 中的选择算子采用轮盘赌选择方式,通过 2.3 节的两个适应度函数(6)、(7)分别计算出种群中每个个体的选择概率。

$$P_1(i) = \frac{f_1(i)}{\sum_{j=1}^{SCALE} f_1(j)} \tag{8}$$

$$P_2(i) = \frac{f_2(i)}{\sum_{j=1}^{SCALE} f_2(j)} \tag{9}$$

选择下一代个体时，首先以 c_1 和 c_2 的概率来分别选择 P_1 和 P_2（其中 $0 < c_1 < 1$，$c_1 + c_2 = 1$），选取其中一个作为选取个体的选择概率。通过这样的选择，种群中即有总任务完成时间较短的个体，又有任务平均所用时间较短的个体，为进化出下一代较优秀的个体提供了优良的基因基础。

2.4.2 交叉与变异操作

交叉是遗传算法中最主要的搜索算子，它模仿自然界有性繁殖的基因重组过程，将原有的优良基因遗传给下一代个体，并生成包含更优良基因结构的新个体。变异可以拓展新的搜索空间，在种群局部收敛时，通过变异可以保持种群多样性。

交叉概率函数和变异概率函数分别为：

$$P_c = \begin{cases} k_1(f_{max} - f')/(f_{max} - f_{avg}), & f' \geqslant f_{avg} \\ k_2, & f' < f_{avg} \end{cases} \tag{10}$$

$$P_m = \begin{cases} k_3(f_{max} - f)/(f_{max} - f_{avg}), & f \geqslant f_{avg} \\ k_4, & f < f_{avg} \end{cases} \tag{11}$$

其中：f_{max} 为群体中最大的适应度值，f_{avg} 为每代群体的平均适应度值，f' 为要交叉的两个个体中较大的适应度值，f 为要变异个体的适应度值。分别用式(6)、(7)得出的两个适应度计算 P_c、P_m，选取其中其较大的作为最终的 P_c、P_m。

3 算法仿真结果与分析

由于云计算的一个局部可以看做一个特殊的网格环境，所以本文用 Gridsim[10] 来模拟一个云计算的局部环境。在相同的环境条件下，分别用自适应遗传算法（Adaptive Genetic Algorithm，AGA）与 DFGA 进行比较。

初始条件：*WORKER* 为 50；*TASK* 为 50；每个 task 划分成子任务数的范围为[20，80]。

算法终止条件为：1）达到最大进化代数 *gnMax*（这里取 *gnMax* = 200）；2）如果连续 50 代总任务完成时间与任务平均完成时间都没有变化时，则认为算法基本收敛，终止算法。

表1 两种算法主要参数

算法	项目	取值	算法	项目	取值
	种群规模	100		种群规模	100
	k_1	0.39		k_1	0.39
	k_2	0.85		k_2	0.85
AGA	k_3	0.096	DFGA	k_3	0.096
	k_4	0.056		k_4	0.056
				c_1	0.7
				c_2	0.3

从图2、图3中可以看出，虽然在进化初期，AGA 得出的总任务完成时间要小于 DFGA 得出的总任务完成时间，而 DFGA 得出的任务平均完成时间要少于 AGA；并且随着进一步的进化，AGA 由于前期只重视总任务的完成时间，因而造成了一些潜在优良基因的丢失，陷入了局部收敛，而 DFGA 随着进化，不但任务平均完成时间少于 AGA，而且在总任务完成时间上也少于 AGA；并且在进化中期，在 AGA 与 DFGA 得出的总任务完成时间相差不大的情况下，DFGA 得出的任务平均完成时间要明显少于 AGA。

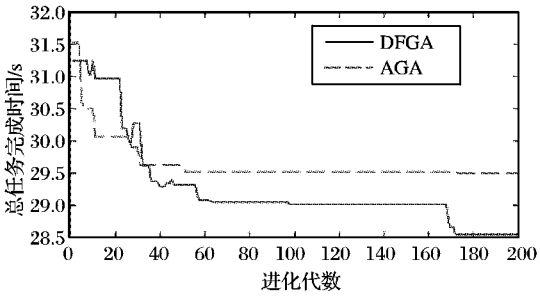

图2 总任务完成时间比较

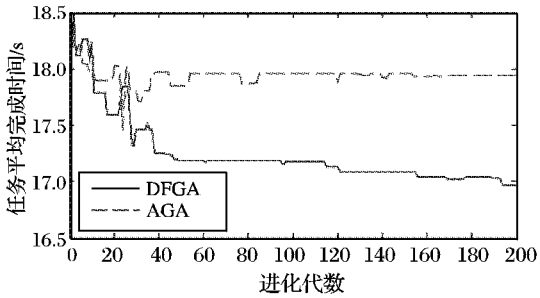

图3 任务平均完成时间比较

4 结语

本文提出了一种基于双适应度的遗传算法的任务调度算法，该算法不但把总任务的完成时间作为一个重要的判断标准，而且也把任务平均完成时间作为一个直接的参考量。通过本算法可以对云计算环境下这种编程模式实现较为理想的任务调度，它是一种有效的任务调度算法。

参考文献：

[1] FOSTER I, YONG ZHAO, RAICU I, *et al.* Cloud computing and grid computing 360-degree compared[C]// Proceedings of the 2008 Grid Computing Environments Workshop. Washington, DC: IEEE Computer Society, 2008: 1-10.

[2] ARMBRUST M, FOX A, GRIFFITH R, *et al.* Above the clouds: A Berkeley view of cloud computing[EB/OL]. [2010-01-25]. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf.

[3] BARROSO L A, DEAN J, HOLZLE U. Web search for a planet: the google cluster architecture[J]. IEEE Micro, 2003, 23(2): 22-28.

[4] 米勒，姜进磊. 云计算[M]. 北京：机械工业出版社，2009.

[5] 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.

[6] KIM J S, NAM B, MARSH M, *et al.* Creating a robust desktop grid using peer-to-peer services[EB/OL]. [2009-10-16]. ftp://ftp.cs.umd.edu/pub/hpsl/papers/papers-pdf/ngs07.pdf.

[7] ABRAHAM A, BUYYA R, NATH B. Nature's heuristics for scheduling jobs on computational grids[C]// The 8th International Conference on Advanced Computing and Communications. New Delhi: Tata McGraw-Hill Publishing, 2000: 45-52.

[8] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters[C]// Proceedings of the 6th Symposium on Operating System Design and Implementation. New York: ACM, 2004: 137-150.

[9] 王小平，曹立明. 遗传算法[M]. 西安：西安交通大学出版社，2002.

[10] The CLOUDS Lab. Gridsim[EB/OL]. [2010-06-25]. http://www.cloudbus.org/gridsim/.