云计算环境下基于用户满意度的遗传算法*

邹伟明，于 炯
（新疆大学 信息科学与工程学院，乌鲁木齐 830046）

摘 要：针对云计算平台的新特征，对原有自适应遗传算法进行改进，提出了一种基于用户满意度的遗传算法（consumer satisfaction genetic algorithm，CSGA）。该算法在保证用户公平性的前提下，将任务调度到输入数据所在的计算节点以减少网络传输开销，并以缩短总任务的完成时间及提高用户满意度为目标优化算法性能。通过仿真实验对比分析 CSGA 与 AGA 算法，实验结果表明该算法在响应时间、公平性和用户满意度方面优于 AGA 算法，更加适应云计算环境。

关键词：云计算；任务调度算法；遗传算法；公平性；用户满意度
中图分类号：TP393 **文献标志码**：A **文章编号**：1001-3695(2014)01-0085-04
doi:10.3969/j.issn.1001-3695.2014.01.019

Consumer satisfaction genetic algorithm in cloud computing

ZOU Wei-ming, YU Jiong
(*College of Information Science & Technology*, *Xinjiang University*, *Urumqi* 830046, *China*)

Abstract: According to the new characteristics of cloud computing platform, this paper proposed an improved genetic algorithm called CSGA. The algorithm under the promise of guarantee consumer fairness, CSGA scheduled tasks to the node with data block of this tasks in order to reduce data translation cost, which aimed to shorten all the task completion time and tried hard to improve the consumer satisfaction. Through the simulation analysis of CSGA and adaptive genetic algorithm (AGA), it shows that CSGA outperforms previous genetic algorithms in term of the job response time and fairness and consumer satisfaction, and the CSGA is better adapted to the cloud computing environment.

Key words: cloud computing; task scheduling algorithm; genetic algorithm; fairness; consumer satisfaction

0 引言

近年来，互联网应用平台（如 Facebook、Google、淘宝等）的数据量日趋庞大，为了解决对海量数据的存储与处理，云计算海量数据处理平台（如 Hadoop[1]、Dryad[2]等）应运而生。作为一种服务模式，为了满足用户“按需使用，按量付费，即需即用”的服务需求，如何在这些云平台上利用有效的资源管理与调度策略提高处理效率成为一个非常重要的问题。

云计算平台用户的多样性决定了其作业类型的多样性，对于数据密集型作业，网络带宽是计算集群中急缺的资源[3]，为了减少任务执行过程中的网络传输开销，可以将任务调度到输入数据所在的计算节点，减少任务的完成时间；而系统中既包括子任务少、执行时间短、对响应时间敏感的即时作业（如数据查询作业），也包括子任务多、执行时间长的长期作业（如数据分析作业），研究公平调度算法可以及时为不同的作业分配资源，使其快速响应以满足客户需求。

本文针对云计算平台的特点，对遗传算法进行改进，提出了一种改进的遗传算法。该算法通过采用公平机制以及数据本地性[4]机制对遗传基因的选择进行约束，以缩短总任务的完成时间和增加用户满意度为双重目标，提高算法的性能，使其更好地适应云计算环境。

1 研究背景

1.1 云计算海量数据处理平台

在云计算海量数据处理平台中，数据文件由云计算分布式文件系统（如 GFS[5]、HDFS[6]等）管理：每一个文件（file）被划分成若干个大小相等的数据块（block），数据块分布于在数据中心计算节点的本地磁盘；为了保证文件的可靠性，每一个数据块都存在多个数据块副本。当处理一个文件时，管理节点将一个作业（job）划分成若干个相互独立的小任务（task），每一个任务处理一个数据块，多个任务可以并行执行，这样可以加速一个作业的完成，当一个作业所有任务完成后，该作业才能完成。如图 1 所示，一个 1 024 MB 的文件被分成八个大小相等（128 MB）的数据块，每个数据块有两个副本，分别存储在不同的计算节点上。为处理该文件，管理节点将该作业分成八个小任务，每个任务处理一个数据块。当某计算节点空闲时，该节点向管理节点请求任务，管理节点根据优先级策略，对队列中的作业排序，选取队首作业中的一个任务，并将其指派到该计算节点。从作业的第一个任务开始执行到最后一个任务结束的时间称为作业的完成时间。当作业中的所有任务都完成后，该作业才能完成。其中从本地磁盘读取输入数据的任务称为数据本地（data-locality）任务，通过网络从其他节点读取输入数据的任务称为数据远程（data-remote）任务。如图 1 所示，计

收稿日期：2013-03-14；**修回日期**：2013-04-22 **基金项目**：国家自然科学基金资助项目（60863003，61063042）；新疆维吾尔自治区自然科学基金资助项目（2011211A011）

作者简介：邹伟明（1987-），男，湖南长沙人，硕士研究生，主要研究方向为网格与云计算（yxzwm3_8@126.com）；于炯（1964-），男，北京人，教授，博士，主要研究方向为网络安全、网格与分布式计算、云计算.

算节点1的数据本地任务为1、2、4、5、7,数据远程任务为3、6、8。

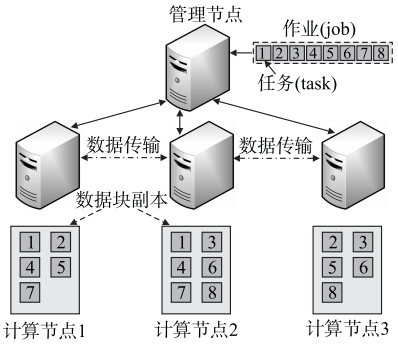

图1 云计算系统体系构架

1.2 遗传算法

遗传算法(genetic algorithm, GA)[7]的基本思想来源于达尔文(Darwin)的进化论和门德尔(Mendel)的遗传学说,由Holland于1975年受生物进化论的启发而提出的。该算法借助于计算机编程,一般是将待求问题表示成串(或称染色体),即为二进制码或整数码串,从而构成一群串,并将它们置于问题的求解环境中,根据适者生存的原则,从中选择出适应环境的串进行复制(reproduction),且通过交叉、变异两种基因操作产生出新的一代更适应环境的串群。经这样一代代不断变化,最后收敛到一个最适应环境的串上,即求得问题的最优解。

遗传算法的实现步骤如下:a)随机产生一组初始群体,群体中的个体(成为染色体)是问题的潜在解; b)计算群体中每个染色体相应的目标函数值(即适应值); c)按一定规则,复制适应值较大的染色体; d)随机地将群体中染色体两两配对交叉,然后以一定的概率使某些染色体产生变异; f)若收敛指标满足则停止,否则转步骤b)。

1.3 问题的提出

遗传算法在作业调度上的应用最早是由Florida大学的Biegel和Davern于1992年提出的。由于云计算平台与以往的计算平台相比又有了一些新特征(如资源同构、MapReduce编程模型、分布式文件系统等),对云计算环境下的任务调度算法具有了新的要求,有必要对原有的遗传算法进行一些改进以适应云计算环境,更好地满足客户要求。为了缩短作业的完成时间及时响应客户需求,以及更好地权衡作业的本地性与公平性,本文对遗传算法作了以下改进:采用公平机制以及数据本地性机制对遗传基因的选择进行约束,增加一个用户满意度适应函数,即把作业的总完成时间及用户满意度作为目标函数(即适应度函数),以此来选择种群。

2 改进的遗传算法

2.1 Max-Min公平算法简介

Max-Min公平算法[8]:已知有 n 个作业 $1,\cdots,n$,每个作业需求的资源数为 $x_1, x_2, \cdots, x_n$。不妨假设 $x_1 \leqslant x_2 \leqslant \cdots \leqslant x_n$。系统的可用资源总数为 C,那么首先分配 C/n 资源给第一个作业即需求最小的作业,再把第一个作业剩余的资源 $C/n - x_1$ 及系统平均分配剩余的资源 $(n-1)C/n$ 平均分配给第二个作业即需求资源第二小的作业。依此类推,直到不能满足某个任务的资源需求,把剩余的所有资源平均分配给不能满足需求的所有任务。

2.2 ETC矩阵及DTC矩阵

ETC(expected time to compute)矩阵[9]:表示任务在不同计算节点上的执行时间,ETC$[i, j]$表示第 i 个任务在第 j 个计算节点上执行完成所要用的时间,该矩阵为常数矩阵。

DTC(data translation cost)矩阵:表示数据传输到不同计算节点所用的时间,DTC$[i, j]$表示第 i 个任务调度到第 j 个计算节点上所需的网络传输开销。若任务 i 的数据块副本存储在计算节点 j 上,则DTC$[i, j] = 0$。该矩阵随远程任务数量的变化动态变化,网络带宽为单位时间内传输的数据量。当远程任务数量增加时,数据的传输开销会线性增大[10],可以表示为

$$\mathrm{DTC}[i,j] = w \times r \tag{1}$$

其中:r 为远程任务的数量;w 为网络因素,w 值越大表示网络越拥塞;$1/w$ 表示网络的数据传输速率。

2.3 染色体编码与解码

染色体编码有很多种方式,可以采用直接编码或间接编码。本文采用资源—任务的间接编码[11]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 J 个作业(job),N 个计算节点(node),第 t 个job划分任务(task)的数量为taskNum(t),则子任务的总数量为

$$\mathrm{sumTaskNum} = \sum_{t=1}^{J} \mathrm{taskNum}(t) \tag{2}$$

假设有三个计算节点,三个作业(job),各个作业分得的任务数量为3、2、5,则总共有10个任务(task),即染色体的长度为10,再对这些任务进行编号,其中较简单的一种编法是,第 i 个作业中的第 j 个任务的序号为

$$m = \sum_{k=1}^{i-1} \mathrm{taskNum}(k) + j \tag{3}$$

编号是1~10,每个基因的取值为1~3,如产生下面的一个染色体:

$$\{1, 3, 1, 2, 2, 3, 2, 1, 3, 1\}$$

则这条染色体代表第1个任务在第1个计算节点上执行,第2个任务在第3个计算节点上执行…… 第10个任务在第1个计算节点上执行。之后就是对染色体解码,得到计算节点上的任务分布情况。生成多组以资源编号的任务序列。如将上述染色体解码为

$$\mathrm{node}_1:\{1,3,8,10\},\mathrm{node}_2:\{4,5,7\},\mathrm{node}_3:\{2,6,9\}$$

通过解码后的序列和ETC矩阵以及DTC矩阵可以计算出每个作业 t 的完成时间以及完成所有作业的总时间。

每个作业 t 的完成时间为

$$\mathrm{jobTime}(t) = \max_{i=1}^{\mathrm{tastNum}(t)} \sum_{j=1}^{q} \mathrm{tastTime}(j,i) \tag{4}$$

即作业 t 中最后一个任务的完成时间,其中 q 为作业 t 中的第 i 个任务被分配到所在计算节点的位置,tastTime(j,i) 为第 i 个任务所在计算节点上完成第 j 个任务所用的时间。

总任务的完成时间为

$$\mathrm{totalTime} = \max_{j=1}^{N} \sum_{i=1}^{p} \mathrm{nodeTime}(j,i) \tag{5}$$

其中nodeTime(j,i)表示在计算节点 j 上完成任务 i 所用的时间(包括任务的执行时间和数据传输时间),p 为分配到计算节点 j 上的任务数量。

2.4 初始种群生成

若种群规模为 S,作业个数为 J,划分的任务总数为 M,计算节点数为 N,则初始化描述为:采用Max-Min公平算法对系统资源进行限制,在公平性约束条件下系统随机产生 S 个染色体,染色体的长度为 M,基因的取值为$[1,N]$。

2.5 适应度函数

遗传算法是通过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,因此适应度的选取至关重要,直接关系到算法的收敛速度与最优解的查找。尽早完成所有任务是任务调度的一个重要目标,而对于即时作业或者交互行作业如何在用户指定的时间内完成以达到用户的满意度,单个作业的完成时间不能忽视,把每个作业的完成时间考虑后,有利于找到实际最优解。因此,本文定义两个适应度函数,保证总任务完成时间短,而且能最大限度地满足用户需求:

a)作业最大截止时间(latestTime)。用户为提交的每个作业设定的最迟完成时间,默认值为 latestTime(t) = taskNum(t) × avgTastTime,其中 avgTastTime 为任务的平均完成时间。若作业 t 能在 latestTime(t)内完成则称作业 t 满足用户要求,即 jobTime(t) < latestTime(t)。

b)用户满意度适应度函数。用户对第 i 个个体中所有作业完成时间的满意程度函数为

$$f_1(i) = \frac{k(i)}{J} \quad 1 \leqslant i \leqslant S \tag{6}$$

其中:$k(i)$ = num((latestTime$_i$(t) − jobTime$_i$(t)) >0),$1 \leqslant i \leqslant S$, $1 \leqslant t \leqslant J$,即第 i 个个体中能在最大截止时间内完成的作业数。

总任务完成时间适应度函数为

$$f_2(i) = 1/CT(i) \quad 1 \leqslant i \leqslant S \tag{7}$$

其中:$CT(i)$表示第 i 个个体完成总任务的时间。

根据以上两个适应度函数选取适应度较大的优良个体遗传到下一代,即用户满意程度越高和总任务完成时间越短的个体越容易被选择。

2.6　遗传操作

2.6.1　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。根据优胜劣汰的原理,个体的适应度越高,被选择遗传到下一代的概率就越大,使得种群中个体的适应度值不断接近最优解。CSGA 中的选择算子采用轮盘赌选择方式,通过两个适应度函数式(6)(7)分别计算出种群中每个个体的选择概率。

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

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

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 $0 < c_1, c_2 < 1$, $c_1 + c_2 = 1$),选取其中一个作为选取个体的选择概率。通过这样的选择,种群中既有用户满意度较高的个体,又有总任务完成时间较短的个体,为进化下一代提供了优良的基因。

2.6.2　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,并生成更加优良基因的新个体。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,防止出现早熟现象。交叉概率函数和变异概率函数分别为

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

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

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,选取其中较大的作为最终的 P_c、P_m。

2.7　改进的遗传算法的实现步骤

改进遗传算法的步骤如下:

a)采用 Max-Min 公平算法对系统资源分配进行约束,在随机产生的群体中选择满足公平性约束条件的个体作为初始群体 S_0。

b)通过群体染色体的编/解码计算出远程任务的数量,并由式(1)计算 DTC 矩阵。

c)根据式(6)(7)计算群体每个染色体相应的适应度函数值。

d)根据式(8)(9)计算群体中每个染色体的选择概率,首先以 c_1 和 c_2 的概率来分别选择 P_1 和 P_2(其中 $0 < c_1, c_2 < 1$, $c_1 + c_2 = 1$),选取其中一个作为选取个体的选择概率,按规则复制概率较大的染色体,得到群体 S_1。

e)按交叉率 P_c 所决定的参加交叉的染色体数 c,从 S_1 中随机确定 c 个染色体,配对进行交叉操作,并用产生的新染色体代替原染色体,得到群体 S_2。

f)按变异率 P_m 所决定的变异次数 m,从 S_2 中随机确定 m 个染色体,分别进行变异操作,并用产生的新染色体代替原染色体,得到群体 S_3。

g)将群体 S_3 作为新一代种群,即用 S_3 代替 S_0,收敛指标满足则停止,否则转步骤 b)。

3　仿真实验结果与分析

本文采用 MATLAB 模拟云计算环境对算法进行仿真实验,在相同环境下对比自适应遗传算法(AGA)[12] 与改进的遗传算法(CSGA),并分析实验结果。

3.1　实验设置

采用 MATLAB 仿真云计算环境,设置 50 个计算节点(即 $N = 50$),每个数据块大小为 128 MB,采用 Hadoop 默认的 3 个副本策略,在 10 Gbit 的网络环境中传输 128 MB 的数据需要 1 s,1 Gbit 的网络环境中需要 10 s,因此在此仿真实验中可设置 $w \in [1, 10]$[10]。

为了验证该算法的普遍适应性,实验将仿照 Facebook 一周提交的数据设置作业的大小,其中 80% 以上的作业为短作业,具体设置如表 1 所示。

表 1　作业参数设置

作业数	任务数	作业所占比例/%	作业数	任务数	作业所占比例/%
19	1 ~ 2	65	3	21 ~ 50	10
6	3 ~ 20	20	2	51 ~ 100	5

总作业数为 30(即 $J = 30$),每个作业划分的任务数为[1, 100]。ETC 矩阵由系统随机生成,两种算法的其他参数如表 2 所示。

表 2　两种算法的基本参数

算法	项目	取值	算法	项目	取值
AGA	群体规模	100	CSGA	群体规模	100
	k_1	0.30		k_1	0.30
	k_2	0.87		k_2	0.87
	k_3	0.10		k_3	0.10
	k_4	0.06		k_4	0.06
				c_1	0.7
				c_2	0.3

算法的终止条件:设置最大进化代数为 200,如果连续 50 代总任务的完成时间和用户的满意度都不变化,则认为算法基本收敛,终止算法。

3.2　实验结果分析

图 2 为各计算节点执行任务数比较。从图 2 可以看出,

CSGA 各计算节点上执行的任务数更加均衡,长作业和短作业都得到了比较合理的计算资源,体现了更好的公平性,而 AGA 算法可能由于计算资源分配不均导致某些节点负载过大,影响作业的完成时间。图 3 为总任务完成时间比较,图 4 为用户满意度比较。从图 3、4 可以看出,在进化初期,AGA 的总任务完成时间要少于 CSGA,而用户满意度要低于 CSGA。随着进化代数的增加,由于 AGA 只以总任务完成时间为目标,难免丢失一些优良基因,而 CSGA 增加用户满意度目标,把每个作业的完成时间考虑进去,因此在进化后期,CSGA 总任务完成时间比 AGA 少,用户满意度也明显高于 AGA,最高可达 96%,而 AGA 的用户满意度在 90% 附近波动。CSGA 的一个不足点是其收敛速率比 AGA 慢,AGA 平均在 50 代左右收敛,而 CSGA 平均 80 代才收敛。

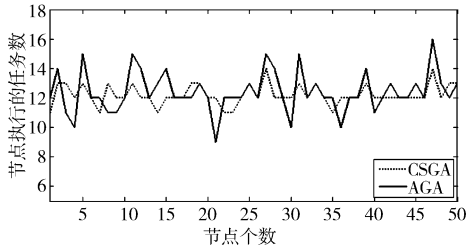

图 2 各计算节点执行任务数比较

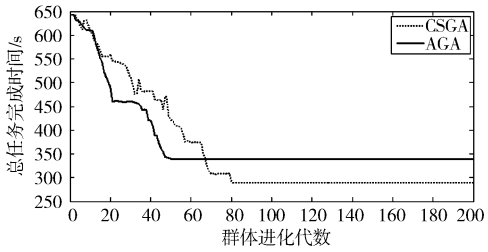

图 3 总任务完成时间比较

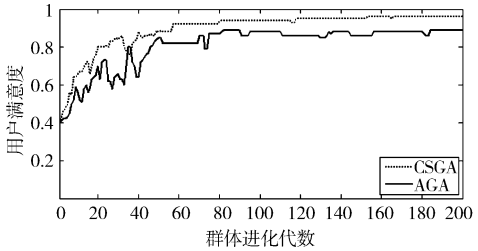

图 4 用户满意度比较

4 结束语

本文提出了一种基于用户满意度的遗传算法。该算法针对云计算平台多用户多作业类型的特征,采用公平算法对计算资源的分配进行限制,以缩短总任务的完成时间和增加用户满意度为双重目标。仿真实验结果表明,该算法体现了良好的公平性,也基本满足了用户要求,是云计算环境下一种有效的调度算法。以后进一步工作的重点是优化用户满意度适应函数,提高算法的收敛速度。

参考文献:

[1] Hadoop [EB/OL]. (2011-12-18) [2012-03-12]. http://hadoop.apache.org.

[2] ISARD M, BUDIU M, YU Yuan, *et al*. Dryad: distributed data-parallel programs from sequential building blocks[C]//Proc of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems. New York: ACM Press, 2007:59-72.

[3] WANG Guo-hui, EUQENE T S. The impact of virtualization on network performance of Amazon EC2 data center[C]//Proc of the 29th Conference on Information Communications. Piscataway, NJ: IEEE Press, 2010:1163-1171.

[4] ZAHARIA M, BORTHAKUR D, SARMA S J, *et al*. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling [C]//Proc of the 5th European Conference on Computer Systems. New York: ACM Press, 2010:265-278.

[5] GHEMAWAT S, GOBIOFF H, LEUNG S T. The Google file system [C]//Proc of the 19th ACM Symposium on Operating Systems Principles. New York: ACM Press, 2003:29-43.

[6] WHITE T. Hadoop: the definitive guide [M]. [S. l.]: O'Reilly Media Inc, 2009.

[7] 王小平,曹立明. 遗传算法——理论、应用与软件实现[M]. 西安:西安交通大学出版社,2002.

[8] Max-Min_fairness[EB/OL]. (2009-12-01)[2013-01-18]. http://en.wikipedia.org/wiki/Max-min_fairness.

[9] BRAUN T D, SIEGEL H J, BECK N, *et al*. A compares on of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems[J]. Journal of Parallel and Distributed Computing, 2001,61(6):810-837.

[10] JIN Jia-hui, LUO Jun-zhou, SONG Ai-bo, *et al*. BAR: an efficient data locality driven task scheduling algorithm for cloud computing [C]//Proc of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. Washington DC: IEEE Computer Society, 2011:1295-304.

[11] 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):184-186.

[12] SRINIVAS M, PATNAIK L M. Adaptive probabilities of crossover and mutation in genetic algorithms[J]. IEEE Trans on SMC,1994,24(4):656-667.