一种适用于大规模变量的并行遗传算法研究

李 东 潘志松
(解放军理工大学指挥自动化学院 南京 210007)

摘 要 当前MapReduce并行编程模型得到了广泛的应用。相对于传统的基于PVM或者MPI的并行编程方式,它在执行时间和处理问题规模等方面有明显优势。针对并行遗传算法的特点,提出基于MapReduce实现一种典型的并行遗传算法——粗粒度并行算法的方法,用以解决大规模变量问题。实验平台采用Hadoop,硬件条件为普通的服务器集群。在多目标优化问题测试中,当问题规模达到一定、处理变量数超过10E+7时,并行算法效率比串行提高数倍,并且能突破内存瓶颈。根据MapReduce自身特点调整其参数,改变并行程度,分析其对并行执行时间的影响。

关键词 大规模变量,MapReduce,并行遗传算法,多目标优化问题,性能分析
中图法分类号 TP18 **文献标识码** A

Research on Parallel Genetic Algorithms Based on MapReduce

LI Dong PAN Zhi-song
(Institute of Command Automation, PLA University of Science and Technology, Nanjing 210007, China)

Abstract MapReduce programming model enjoys widespread application and becomes new popular parallel programming paradigm nowadays. It uses MapReduce model to parallelize coarse-grained genetic algorithms, and takes mult-objective optimization problem as the benchmark. All the experiments are made under hadoop and a cluster which consists of commodity servers. When the number of variable reach to 10E+7, the efficiency of parallel algorithm can be multiplied several times without introducing any bottlenecks revolved memory. Finally we studied how the parallel degree affects the performance.

Keywords Large-scale variables, MapReduce, Parallel genetic algorithms, Mult-objective optimization, Performance analysis

1 引言

遗传算法(Genetic Algorithm)是美国Holland教授于1975年正式提出的,它通过当前群体模拟生物进化的方法产生下一代。主要的遗传操作包括变异和交叉[1]。Holland教授在提出遗传算法之初,就指出这种算法存在先天的并行性。并行遗传算法(Parallel Genetic Algorithm)正是基于遗传算法内在的并行特征提出的。一般根据并行的粒度,即分而治之的方法应用到遗传算法的不同层次、不同方面,可以大致将并行遗传算法分为4类[2]:全局型的,或称主从式的;单种群细粒度;多种群粗粒度的;混合型的。

传统的基于MPI或者PVM的并行遗传算法实现需要计算机体系结构方面的知识,实现起来较为复杂。并且,随着问题规模的增长,通信所用时间占程序执行时间比例增大。

MapReduce[3]是Google在2004年公布的分布式并行计算模型,用户通过指定Map操作处理键值对(key/value)并产生中间结果,再通过指定Reduce操作将相同的键所对应的值规约起来形成结果。这种模型允许用户编写的程序运行在由廉价普通机器组成的大规模集群之上,并在可编写的程序多样性和执行性能方面表现出色。

本文的主要目的是用MapReduce来实现一种典型的并行遗传算法,并对其性能进行分析。本文第1节为引言;第2节绍国外相关工作及研究现状;第3节描述算法的实现;第4节为实验及结果分析;最后为结束语。

2 研究现状

自MapReduce提出以来,不断有探讨传统并行算法实现向MapReduce移植的文献诞生。就比较有代表性的机器学习领域,文献[4]给出了很多机器学习算法并行化的一般过程。文献[5]实现了MapReduce与粒子群算法的结合。文献[6]则是探讨了用MapReduce实现并行决策树集成算法。

文献[7,8]分别在不同方面实现了MapReduce与并行遗传算法的结合,并就各自特点给出了一些结论。文献[8]指出MapReduce本身不适合遗传算法的表述,因此多加了一个Reduce过程。在Map和Reduce阶段,它使用的键多为1,失去了MapReduce自身的优势;其实现基于.NET平台,不同于流行的Hadoop[9]基于Java平台。Hadoop项目是Apache基金会下的关于Google MapReduce和Google FileSystem等相关技术的开源实现,也是目前MapReduce众多实现版本中的事实上的标准[6]。

文献[7]则采取不同策略,依靠MapReduce自身特点,定制出适合该编程模型的算法,充分发掘MapReduce的并行优

到稿日期:2011-08-25 返修日期:2011-11-22 本文受国家自然科学基金(60603029)资助。
李 东(1987—),男,硕士生,主要研究方向为模式识别、人工智能;潘志松(1973—),男,教授,主要研究方向为网络安全、模式识别。

势，解决了一个大规模变量的问题。

3 基于 MapReduce 的并行遗传算法实现

第 1 节介绍的几种不同类型的并行遗传算法，其提出和实现跟当时的计算技术密切相关。如全局型并行遗传算法，其实现多基于 PVM 并行计算环境，这是一种易于用主从节点分布式实现的计算模型；细粒度模型，其网格状的拓扑结构与共享内存多处理机或分布式内存计算机的体系结构息息相关，事实上其实现也是在这些计算机上实施的[2,10]；粗粒度模型应用最为广泛，已有前人实现的 MPI 或 PVM 版本。这些技术的好处是可以在单机上模拟并行，也可以在集群上实际并行。但 MPI 基于消息的机制实现起来难度较大，需要程序员考虑的通信细节较多，最重要的是，MPI 在问题规模增长的情况下，通信时间占到总的运行时间比例显著增加，导致程序效率低下[11]。

能够克服 MPI 程序的这些局限，正是 MapReduce 的优势所在，后者表现出的可伸缩性(scalable)越来越得到更多人的认可。就几种并行遗传算法的特征来看，粗粒度模型最容易通过 MapReduce 实现，也最直观。

种群初始化后，每一代进化的过程用一个 MapReduce 来完成。其中 Map 完成适应度评价，将子群编号作为键，个体作为值。这一部分通常比较耗时[7]，可以并行操作(见图 2)；Reduce 将相同的键对应的值归约起来，完成一个亚种或者子群的选择、交叉、变异(见图 3)。这样就保持了子群进化的相对独立性。同时，设置多个 reducer，亚种进化也是并行地进行。迁徙可以在 Map 阶段完成，通过一定的策略改变某个个体的键即子群编号，达到该个体从一个子群迁徙到另外一个子群的目的。迁徙是并行遗传算法改变串行遗传算法的一个重要方面，前人已有文献研究迁徙的比例、时机、拓扑等重要问题。但关于它能在多大程度上影响最终结果，尚无一般结论[2]。这里只采用一种随机的迁徙策略。从这个角度来看，若论最终结果(如找到最优解、收敛到何种程度等)，并行遗传算法未必能找到比串行算法更好的结果。并行的优势在于时间以及问题的规模上，第 4 节将会详述。算法整体模型如图 1 所示。

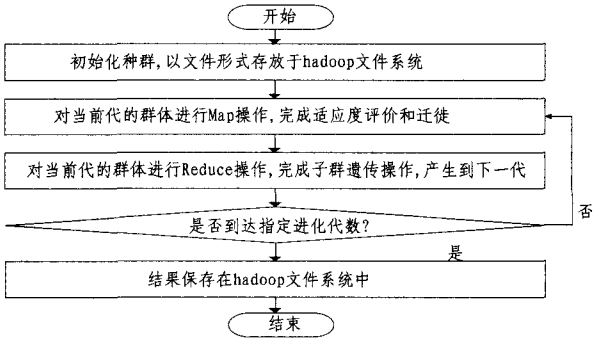

图 1 算法整体运行流程图

```
Map 阶段
Map(key,value)
fitness=Fitness(Value);
value=bind(fitness,value);
if(满足迁徙条件)
  key=newkey(key)
write(key,value)to DFS
End
```

图 2 Map 操作

```
Reduce 阶段
Reduce(key,values)
For each value in values
  fitness[i]=value. getfitness();
  individuals[i]=value. getvalue();
  Maxfitness=MAX(Maxfitness,fitness[i]);
end
  按照轮盘赌的方式选择更新子种群，同时采取精英保留策略。
individuals=Select(individuals);
分别按 r 和 m 的概率对整个子群执行交叉和变异
individuals=Crossover(individuals);
individuals=Mutate(individuals);
将每个新个体按照(key,value)写入 DFS
For each individual in individuals
  Write(key,individual)
end
End
```

图 3 Reduce 操作

从上述算法流程及 MapReduce 操作来看，并行程序的最大优势在于 Map 和 Reduce 并行操作，节省了时间。根据加速比的一般公式，即串行程序执行时间比并行程序执行时间，计算每一代时间加速比为

$$\rho=\frac{F(n)+G(n)}{F(\frac{n}{m_1})+G(\frac{n}{m_2})+C(n)} \tag{1}$$

式中，n 为种群规模，$F(n)$为适应度值计算时间，$G(n)$为遗传操作时间，m_1，m_2 分别是同时运行 Map 和 Reduce 的数量，即并行度。$C(n)$为通信传输时间，主要是种群从 Map 到 Reduce 经过网络传输的时间；F 和 G 都与种群规模成正比，即

$$F(\frac{n}{m_1})=m_1F(n),G(\frac{n}{m_2})=m_2G(n) \tag{2}$$

取

$$m_1=m_2=m,T(n)=F(n)+G(n),\frac{T(n)}{C(n)}=k \tag{3}$$

则

$$\rho=\frac{km}{k+m} \tag{4}$$

式中，k 为计算通信比，m 为并行程度。式(4)表明，并行加速性能主要受这两个因素影响。当 $k\gg m$ 时，加速比接近于 m，即网络开销小；大都是本地计算时，加速比主要由并行程度主导；当 $m\gg k$ 时，加速比接近于 k，即网络开销大时加速比受制于计算通信比；更一般的情况介于两者之间。

4 实验结果

多目标优化问题(MOP)常用作演化算法的标准测试问题。一般来讲，多个目标不能同时达到最优，而是互相牵制，形成一组均衡的解，称为最优非劣解集，也叫 Pareto 解集。这些解对应的目标向量称为 Pareto 最优前沿。更多关于 MOP 构造和 Pareto 最优前沿方面的内容见文献[12,13]。

实验采用文献[13]提出的双目标优化问题，可以明确知道 Pareto 最优前沿，方便验证。遗传操作中的适应度评价就是基于 Pareto 最优前沿的，即当前个体距 Pareto 最优前沿的欧式距离，这也是早期 MOEA 的基本作法。具体公式如下所示，距离越小，适应度越大：

$$F(x)=\frac{1}{1+\|x-y\|} \tag{5}$$

式中，x 为当前个体，y 为离 x 最近的 Pareto 最优前沿集中的相对应个体。这里为计算简便，找出 y 的原则并不严格按照几何最近的方法，而是取出对应自变量相等的因变量值。具体测试函数为 ZDT1，定义如下：

$$\begin{aligned}&\min f_1(x_1)=x_1\\&\min f_2(x_2,x_3,\cdots,x_m)=1-\sqrt{f_1/g}\\&g(x_2,x_3,\cdots,x_m)=1+9\sum_{i=2}^{m}x_i\\&x_i\in[0,1]\end{aligned}\tag{6}$$

其 Pareto 最优前沿为 $g=1$ 时，如图 4 所示。

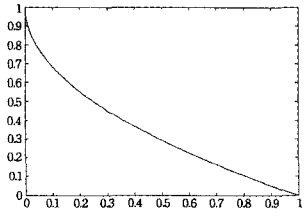

图 4　ZDT1 的 Pareto 最优前沿

实验硬件条件为 Dell PowerEdge 服务器。软件平台为 Java 和 Hadoop。单机串行程序运行在 Dell PowerEdge R910 上，16 核 Intel Xeon 处理器 16G 内存。集群分两种：前期采用 2 台 Dell PowerEdge 服务器作为受控节点，每台包括 24 核 Intel Xeon 处理器 64G 内存；后期采用 8 台普通服务器，每台包含四核 Intel Core i7 处理器和 8G 内存。随着变量数不断增加，对比串行算法运行时间与 MapReduce 运行时间，如表 1 所列。

表 1　串行和并行运行时间对比(单位：s)

个体数	串行时间	并行时间	Map时间	Reduce时间	新集群并行	两集群时间比	并行多源输出
10000	0.4	27	9	15	22	1.136	11
50000	1.67	37	12	24	24	1.583	11
100000	3.33	27	10	15	25	1.17	12
150000	5.17	27	9	15	27	1.0	12
500000	18.24	39	12	24	32	1.218	13
750000	26.85	45	15	27	38	1.184	13
1000000	42.10	51	15	33	40	1.275	14
1200000	45.99	54	15	36	44	1.227	14
1400000	56.49	60	15	42	48	1.250	15
2000000	78.85	76	19	54	57	1.316	15
3000000	120.06	95	20	70	72	1.319	18
4000000	163.26	111	21	87	92	1.206	20
5000000	213.11	135	28	105	104	1.298	22
6000000	300.54	156	30	123	123	1.268	26
7000000	372.14	178	30	145	137	1.299	27
8000000	395.53	197	30	165	156	1.263	30
9000000	430.46	223	37	181	171	1.304	34
10000000	519.19	259	40	209	191	1.335	36
20000000	1079.53	502	96	396	407	1.233	60
30000000	超出内存	681	102	571	564	1.207	75
40000000		1029			675	1.524	110
50000000		1174			824	1.425	123
60000000		1465			1001	1.466	134
70000000		1694			1148	1.476	154
80000000		1956			1410	1.387	172
90000000		2237			1476	1.515	190
100000000		2507			1634	1.534	220

其中，优化后的多源输出模型是基于 Reduce 时间过长、并行程度不够提出的。将计算向 Reduce 倾斜，同时改变单一大文件追加写的弊端，可大大提升程序运行速度。直观起见，串行执行时间和并行时间对比如图 5 所示。

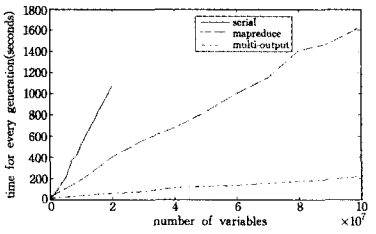

图 5　串行和并行时间对比

图 5 中，纵坐标代表平均每代进化所需时间，其中串行执行时间为 10 代平均值。并行执行时间每代为一个 MapReduce 作业，随机取出一个执行时间。多源输出的每代执行时间较为稳定。当变量数小于一定值时，串行程序执行时间较短，这是因为 MapReduce 作业建立以及分布式系统通信要消耗一定时间；当变量数超过 2.0E+06 时，未经优化的串行时间超过并行时间，且上升更快。变量数超过 3.0E+07 时，单机由于内存限制，无法处理；并行程序则表现出良好的伸缩性，运行时间与变量规模几乎呈线性关系，且能处理更大规模的问题，这种特性随着多源输出的引入变得更加明显。程序运行中通过观察资源消耗发现，串行程序尽管运行在多核的单机上，但只要一个 CPU 在使用，就能很快达到饱和；而并行程序几乎均匀地使用了多个处理器。

并行程序的执行时间取决于并行的程度。就 MapReduce 来说，并行程度反映多少个 Mapper 和 Reducer 同时运行。集群 1 包含最大 Mapper 数为 40，集群 2 为 56。表 1 最后一列表示时间对比。图 6 反映了两个集群运行的时间。

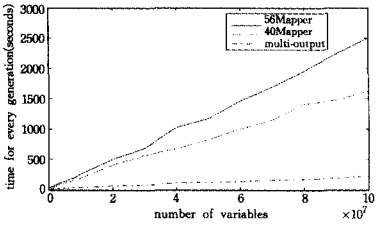

图 6　两个集群的运行时间对比

可以看出，集群 2 的并行程度更高，因此执行时间更短。同时，在同一规模变量下，两者时间之比集中在 1.2～1.5 之间，与其 Mapper 数之比 1.4 对应；同时将 Reducer 并行程度加大时，运行时间缩短得更多。说明了在一定规模变量下，MapReduce执行时间直接受限于并行程度。

结束语　上述结果说明了 MapReduce 在处理大规模数据方面相对于传统串行算法的优势。为结合 MapReduce 与遗传算法，笔者曾尝试过将旅行商问题(TSP)作为测试函数，因为前人有很多的关于遗传算法及其变体解决 TSP 问题的论述[14,15]。但结果显示，若不对遗传算子本身进行改进，仅靠 MapReduce 并行是不能取得令人满意的结果的。尤其是在城市数增大的情况下，MapReduce 解决大规模问题的优势不能脱离基于它的原串行算法的性能。目前解决大规模 TSP 问题的主流方法是启发式算法[16]。遗传算法作为一种基于局部搜索的办法，可能不是解决这个问题的最佳途径。

将 MapReduce 实现并行遗传算法的性能与 MPI 等传统并行编程方法实现的性能做对比，可能是一个有意义的话题。

(下转第 204 页)

认为缺失角色填充正确。还有关于深层语义关系的问题，如事件发生的地点为“北京”，一般认为此事件发生在“中国”也是正确的。然而，在语料中，“北京”和“中国”两个实体编号不同，故在我们的评测中，认为此类填充是错误的(负样例)。

结束语 本文提出了缺失事件角色填充理论并实现了基准本台。实验的识别和分类阶段分别给出了基本特征集、实验过程和结果，两个阶段的 F 值分别达到 72.97 和 74.68。

对于缺失事件角色填充理论的研究，本文做了初步探索。目前，事件抽取总体性能不高，我们的研究基于 ACE2005 Golden 语料，下一步将结合事件抽取研究二者的联合学习方法。另外，缺失事件角色填充理论基于单文档的跨事件理论，下一步将扩展到跨文档领域。

参 考 文 献

[1] ACE Chinese Annotation Guidelines for Events [R]. http://www.ldc.upenn.edu/Projects/ACE/docs/Chinese-Events-Guidelines_v5.5.1.pdf. 2005c

[2] Grishman R, Westbrook D, Meyers A. NYU's English ACE 2005 System Description[R]. 2005

[3] Ahn D. The stages of event extraction [C]// Proceedings of the Workshop on Annotations and Reasoning about Time and Events. Sydney, July 2006:1-8

[4] Ji Heng, Grishman R. Refining Event Extraction through Cross-Document Inference[C]// Proceedings of ACL. HLT, Columbus, 2008:254-262

[5] Gupta P, Ji Heng. Predicting Unknown Time Arguments based on Cross-Event Propagation[C]// Proc. ACL-IJCNLP. 2009:369-372

[6] 许荣华，等. 基于指代消解得中文事件融合方法[J]. 计算机应用，2009，08(1)：2264-2267

[7] Huang, Riloff. Peeling Back the Layers: Detecting Event Role Fillers in Secondary Context[C]// Proceedings of the 49th Annual Meeting of ACL. 2011

[8] Liao Sha-shaa, Grishman R. Using Document Level Cross-Event Inference to Improve Event Extraction[C]// Proc. ACL. Uppsala, Sweden, 2010:789-797

[9] Ji Heng. Challenges from information extraction to information fusion[C]// Proceedings of COLING. Beijing, China, 2010:507-515

[10] Zhao S H, Ng H T. Identification and Resolution of Chinese Zero Pronouns: A Machine Learning Approach[C]// ACL'2007. 2007:541-550

[11] Kong F, Zhou G D, Zhu Q M. Employing the Centering Theory in Pronoun Reslution from the SemanticPerspective[C]// EMNLP'2009. 2009:987-996

[12] 赵妍妍，秦兵，车万翔，等. 中文事件抽取技术研究[J]. 中文信息学报，2007，22(1)：3-8

[13] Chen Zheng, Ji Heng. Language specific issue and feature exploration in Chinese event extraction[C]// Proceedings of NAACL HLT. Boulder, Colorado, USA, 2009:209-212

[14] 付剑锋，刘宗田，付雪峰，等. 基于依存分析的事件识别[J]. 计算机科学，2009，36(11)：217-219

(上接第 184 页)

另外，MapReduce 参数调优以提高性能，以及尝试用 MapReduce 并行遗传算法解决更为实际的问题并对此做出更深入的理论探讨，也是笔者未来的工作之一。

参 考 文 献

[1] Mitchell T M. Machine Learning[M]. 曾华军，张银奎，等译. 北京：机械工业出版社，2003

[2] Cantú-Paz E. A Summary of Research on Parallel Genetic Algorithms[J]. IlliGAL R, 1997(3)

[3] Dean J, Ghemawat S. Mapreduce: Simplified data processing on large clusters[C]// Operating Systems Design and Implementation. 2004:137-149

[4] Chu C-T, Kim S K, Lin Yi-an, et al. Map-Reduce for machine learning on multicore[C]// Neural Information Processing Systems 19 (NIPS 2006). Vancouver, British Columbia, Canada, 2006:281-288

[5] McNabb A W, Monson C K, Seppi K D. Parallel PSO Using Map-Reduce[C]// Proc. of the Congress on Evolutionary Computation. Singapore, 2007

[6] Panda B, Herbach J S, Basu S, et al. PLANET: Massively parallel learning of tree ensembles with MapReduce[C]// Proc. of the 35th International Conference on Very Large Data Based(VLDB 2009). Lyon, France, 2009:1426-1437

[7] Verma A, Llora X, Goldberg D E, et al. Scaling genetic algorithms using mapreduce[C]// InInternational Conference on Intelligent Systems Design andApplications. Pisa, Italy, 2009

[8] Jin C, Vecchiola C, Buyya R. Mrpga: An extension of mapreduce for parallelizing genetic algorithms[C]// IEEE Fourth International Conference on eScience'08. 2008:214-221

[9] http://hadoop.apache.org/

[10] 王竹荣，巨涛，马凡. 多核集群系统下的混合并行遗传算法研究[J]. 计算机科学，2011，38(7)：194-199

[11] Qiu J, Ekanayake J. Thilina Gunarathne, et al. Data Intensive Computing for Bioinformatics[EB/OL]. http://grids.ucs.indiana.edu/ptliupages/publications/DataIntensiveComputing_BookChapter.pdf, 2010

[12] Deb K, Thiele L, Laumanns M, et al. Scalable multiobjective optimization test problems[C]// Proc. of Congress on Evolutionary Computation. 2002

[13] Deb K. Multi-objective genetic algorithms: Problem test Functions[J]. Evolutionary Computation, 1999, 7(3), 205-230

[14] 王斌，李元香，王治. 一种求解 TSP 问题的单亲遗传算法[J]. 计算机科学，2003，30(5)：73-75

[15] 蔡之华，彭锦国，高伟，等. 一种改进的求解 TSP 问题的演化算法[J]. 计算机学报，2005，28(5)：823-828

[16] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/