MPI环境下的几何定理并行自动推理

潘 斌[1]，郭红霞[2]

(1.成都理工大学信息管理学院 成都 610059；2.成都大学电子信息工程学院 成都 610106)

【摘要】将几何定理机器证明和并行计算结合起来考虑，尝试用并行计算方法来提高传统定理证明算法效率，探讨了前推法、数值并行法的并行算法，分析了两种定理证明算法在消息传递编程模型下的任务划分、通信组织、任务调度等问题，并用MPICH2实现了这两种并行算法，对算法的并行性能指标进行了测试，测试数据表明，两种并行算法在基于MPI-2的并行计算环境下，能很好地发挥并行计算的优势，有效缩短构造性几何命题机器证明的时间。

关 键 词 前向推理； 并行算法； 数值并行法； 性能量度； 定理证明

中图分类号 TP311 文献标识码 A

Parallel Automated Reasoning for Geometry Theorem Proving Based on MPI Environment

PAN Bin[1] and GUO Hong-xia[2]

(1. College of Information Management, Chengdu University of Technology Chengdu 610059;

2. College of Electronic and Information Engineering, Chengdu University Chengdu 610106)

Abstract This paper describes two parallel algorithms for geometry theorem proving based on the two traditional methods: the forward reasoning and the numerical verification method. The task partitioning, communication, and the task-scheduling algorithm are also described with the message-passing programming model. Tests on the parallel computing environment are reported. The results demonstrate that proving time of the program is shorten effectively.

Key words forward reasoning; parallel algorithm; parallel numerical method; performance metrics; theorem proving

人工智能与自动推理技术的应用十分广泛，几何定理机器证明是其中一个重要的分支。在实际运用中，对很多复杂度较高的问题，现有的算法在合理的时间内还不能解决。20世纪末以来，国外的一些研究人员已经开始尝试将并行计算技术引入到符号计算当中，以解决一些高复杂度的问题[1]。本文将几何定理机器证明和并行计算结合起来考虑，主要探讨前推法、数值并行法的并行化算法，并在消息传递接口MPI-2的一个具体实现即MPICH 2所构建的并行计算环境下进行实现和测试。

1 理论背景

前推法是一种基于规则的推理方法，类似于人的解题过程。前推法有两个特点：

(1) 能够产生传统形式的可读证明；

(2) 无论结论是否能够被推出，都能产生大量有用信息。但是，当前推达到推理不动点仍未推出命题结论，并不能断定命题不成立，需要用其他方法辅助推理，如添加辅助线、使用反证法等。实际上，可以用代数的方法(吴方法、数值验证法等)对命题的真伪做预先判定。

数值验证法的基本思想是：要肯定或否定一条初等几何命题，只要检验若干个数值实例即可。与吴方法等代数方法相比，该方法用数值计算代替符号计算以减少解题难度，是很有特色的一种辅助方法。

文献[2-3]介绍了前推法的基本理论，关于数值验证法的具体理论阐述，可参看文献[4-5]。

2 并行算法设计

2.1 并行前推法

并行前推法程序从文件中读入一个几何命题，将题设条件写入初始信息库，并不断将规则库中的

收稿日期：2007−04−08; 修回日期：20008−01−24
基金项目：国家973计划基金资助(CB318003)
作者简介：潘 斌(1978 –)，男，博士生，主要从事MIS、并行工程、专家系统方面的研究.

规则应用于信息库，直到无法产生新的几何信息，然后检查信息库中是否存在命题结论，若存在，则输出可读证明到文件中，否则，打印命题证明失败的信息。

(1) 任务划分。对各子步骤的数据依赖关系作分析，结果如图1所示。

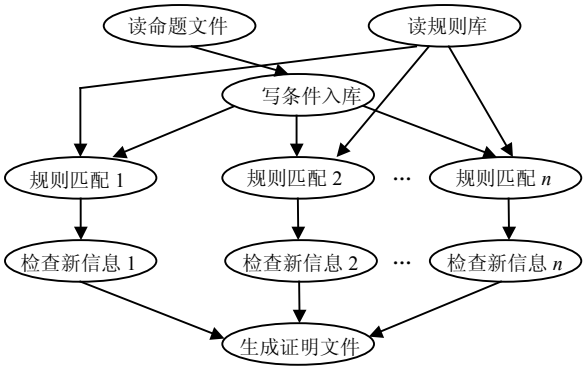

图1　并行化后的数据依赖图

于是，每个操作可以映射为一个原始任务，每个原始任务和2个数据单元(条件信息组+规则)相关联。比较好的数据关联策略是所有计算节点采用同样的规则集，不同的条件信息组进入节点进行匹配，这是条件组层次的并行，其并行粒度小，可扩展性也更好。

由于任务的数目在编译时无法确定，各个任务之间不需要通信，完成不同任务所花费的时间也不尽相同，所以任务的映射应该在运行期完成，本文选择Master-Slave模式[6-7]实现算法，用Master进程进行I/O操作和任务分发。

(2) 通信组织。本文采用任务/通道模型[7]来描述算法中的通信情况。最初的通信由Slave进程开始，它发送一个“就绪”消息给Master进程，Master进程用题设条件构成初始信息库，向所有Slave进程广播该信息库，然后开始构造和分发规则匹配任务。在图2中，细虚线箭头表示广播规则的通道，粗虚线箭头表示信息库广播和匹配任务分发的通道，点划线箭头表示新信息回送给Master的通道。

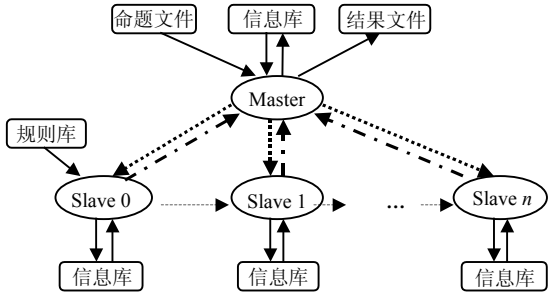

图2　前推法任务/通道图

在进行一轮规则匹配后，各个Slave进程产生的新信息是不一样的，需要进行调整以保证本地和全局信息库的一致性。显然，Master进程收集完新信息后，只需向各Slave进程广播一次即可。

(3) 任务调度和复杂度分析　Master-Slave程序模式采用的是一种动态的任务分配策略，为了减少进程间额外的通信开销，Master进程在给Slave进程分配数据的时候，使用按组分配任务的策略，每次分配k个任务。k的具体数值受多个因素的影响，需要进一步分析。

先前的设计中，构造匹配任务被设计成串行任务。本文采用流水线法进行改进：当Slave处理已有的匹配任务时，Master进程可以同时开始构造新的匹配任务。

但是需要注意的是，如果当前信息库中的信息都已经用来产生匹配任务，那么Master进程停止匹配任务的构造，等待信息库的更新。

本文仅粗略分析算法主体部分(忽略文件I/O时间)执行时间的表达式。由于采用了流水线方法，Master进程上进行的匹配任务的串行时间只需要计算到生成第一批任务的时间。

设N为一次证明产生的匹配任务总数，P为处理器数目，k为每个处理器完成一轮推理的任务数目，λ为消息传递的延迟时间，t_0为Master进程产生一个匹配任务的时间，t_1为Slave进程完成一次规则匹配的时间(包括检验新信息)。在一轮推理中，要进行信息库分发和结论信息收集2次广播通信。由于在MPI中，实现广播通信的算法的通信步数为$\lceil \mathrm{lb}P \rceil$[8]，所以总的开销为$2\lambda\lceil \mathrm{lb}P \rceil$，还要进行一次任务的分发，其开销为$\lambda P$，所以算法的执行时间为：

$$(kP)t_0 + \frac{N-kP}{kP}(kt_1 + 2\lambda\lceil \mathrm{lb}P \rceil + \lambda P)$$

式中　$(kP)t_0$为Master产生第一批匹配任务的时间；$N-kP$是剩余的匹配任务；$\frac{N-kP}{kP}$为循环推理的次数；kt_1为单个处理器完成每一轮规则匹配的时间(完成k个任务)。

实际上，对每一个几何命题，N是不能预先确定的，并且，虽然把每次消息传递的延迟统一处理了，但还是要受信息库以及推理结果信息大小的影响，所以上述表达式只能作为复杂度分析的参考，算法效率的分析应该以实际运行指标为重点。

从上式还能够看出，当$k=\sqrt{\frac{2N\lambda(\lceil \mathrm{lb}P \rceil + P)}{(t_0-t_1)P^2}}$时，时

间表达式有最小值，这也从理论上回答了前面的k值确定问题。

2.2 并行数值验证算法

并行数值验证程序从文件中读入一个几何命题，将题设条件转化为多项式方程组的形式，确定验证实例的规模，然后对各实例在结论中是否成立进行并行验证，如果存在不成立的实例，则输出命题不成立的信息；否则，待实例全部验证完毕，输出命题成立的信息。

(1) 任务划分。考虑到实例验证占用了大部分运行时间，而且各个实例验证时是互相独立的，任务划分可以使用功能分解：每个验证操作映射为一个原始任务，每个原始任务和一个实例相关联。由于算法开始阶段还存在一些串行操作，如读取命题文件、确定任务规模等，结束阶段要输出结果信息，本文仍然选择Master-Slave模式来实现算法。

(2) 通信组织。在图3所示的数值并行法任务/通道图中，细虚线箭头表示广播命题信息的通道，粗虚线箭头表示实例验证任务分发的通道，点划线箭头表示结果信息回送给Master的通道。

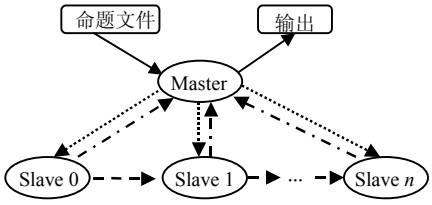

图3 数值并行法任务/通道图

(3) 任务调度和复杂度分析。由于任务总数能够预先确定，所以可以采用静态的任务调度策略，一次产生全部任务，然后分发。但是对于每个验证任务来说，其时间复杂度和分配到的数值大小有很大关系，采用静态的任务调度，很可能造成计算节点负载不均衡，影响算法效率。为了平衡节点负载，可以考虑采用任务池竞争的分配方法来实现：Master进程不断生成任务写入任务池(可以是共享内存也可以是文件)，Slave进程完成一组任务后，从任务池中取另一组任务，直到任务池为空，这就保证了各Slave进程始终处于工作状态。但是，采用任务池进行任务调度会明显增大进程间的通信开销，与静态的调度方法相比孰优孰劣，还需进一步分析[9]。

忽略任务开始阶段的广播开销，在静态调度方法中，任务生成的总时间为 Nt_0，任务分发是一次完成平均分配，每个处理器完成验证的总时间为 $\frac{N}{P}t_1$，故总的耗时为：

$$T_1 = Nt_0 + \lambda + \frac{N}{P}t_1$$

在任务池方法中，由于采用了流水线处理，任务生成的串行时间仅需要计算到前kP个任务。取任务大概需要进行N/kp次，每个处理器完成k个任务需要一次取任务的通信开销，故总的耗时为：

$$T_2 = kPt_0 + \frac{N}{kP}(\lambda P + kt_1)$$

易知，当 $k = \sqrt{\frac{\lambda N}{Pt_0}}$ 时，上式有最小值：

$$T_3 = 2\sqrt{\lambda NPt_0} + \frac{Nt_1}{P}$$

为了比较T_1、T_3的大小，令 $T_4 = T_1 - T_3 = Nt_0 - 2\sqrt{\lambda NPt_0} + \lambda$，由于只关心$N$的不同取值对效率的影响，故把$N$看作主变量，注意到$N$>0，易求得当 $\sqrt{N} > \frac{\sqrt{\lambda}(\sqrt{P}+\sqrt{P-1})}{\sqrt{t_0}}$ 时，$T_4 > 0$，即 $T_1 > T_3$。

实际上，t_0 是相当小的，在本系统中测得 $t_0 \approx 0.000\ 6$ s；$\lambda \approx 0.008\ 5$ s。代入上式，若P=10，那么当N>567时，任务池调度较优；若P=100，那么当N >566 7时，任务池调度较优。所以，在处理复杂的命题时，采用任务池的调度方法具有较好的效率。

3 算法测试

由于影响并行算法效率的因素较多，在具体的计算环境中进行试验是检验算法效率必不可少的步骤。本文所作的算法测试是在以MPICH 2组建的并行计算环境下进行的。

通过对32个难易不同的几何命题进行验证，表明算法的并行设计比较成功，下面就以其中有代表性的3例进行分析，评价指标采用并行加速比和并行效率[10]。

例1 如图4所示，在以O为圆心的△ABD的外接圆上取一点C，AD、BC交于点E，△DCE的外接圆和△ABE的外接圆交于点E和F。求证：OF⊥EF。

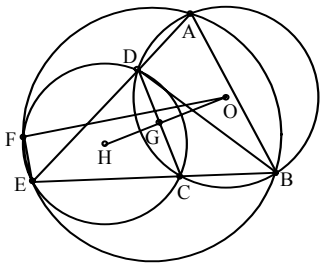

图4 例1

例2 如图5所示，设BD是直角三角形ABC斜边上的高，E是BD的中点，过E作AB、BC、AC的平行线分别与AC、BC交于F、G，与AB、AC交于H、I，与AB、BC交于J、K。求证：G、H、K、J 这4点共圆。

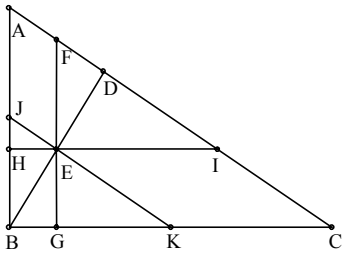

图5　例2

例3 如图6所示，△ABC的三垂线AD、BE、CF分别交BC、AC、AB于D、E、F，且AD、BE、CF又相交于H，HP⊥FE、HQ⊥FD。求证：PQ//AB。

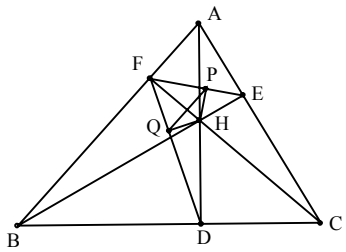

图6　例3

实际测得两种并行算法的运行时间如表1、表2所示，表中的时间单位为秒。

表1　并行前推法执行时间

节点数	例 1	例 2	例 3
单机	80.22	5.39	24.71
2	42.67	5.08	13.88
3	30.74	5.23	10.13
4	24.09	5.34	8.73
5	21.62	5.67	7.58
6	18.96	7.38	7.33
7	17.51	7.19	6.90
8	15.70	7.93	6.67

表2　数值并行法执行时间

节点数	例 1	例 2	例 3
单机	1 818.74	50.78	1.75
2	927.93	26.73	1.17
3	635.92	18.81	1.26
4	490.23	14.94	1.41
5	417.14	13.02	1.51
6	348.59	11.54	1.59
7	313.58	9.84	1.66
8	281.98	9.07	1.70
任务规模	509 796	33 856	256

依据表1，计算并绘制并行前推法的加速比曲线，如图7所示。图7中的虚线是加速比的估计值，它根据Gustafson-Barsis定律[11]计算。计算时串行时间比例取为0.1。

从图中可以看出，并行前推所能获得的加速比和并行效率并不十分理想，8节点下最大加速比5.11也与估计值7.3相差甚远。在证明例2时，大于4个节点参与计算时，节点间的通信开销已经超过了任务分摊所能减少的执行时间，这是由于推理时产生的总任务数不多。但是当单节点计算耗时较多时(如例1)，并行算法还是能显著减少解题时间。这种情况是算法和系统的并行开销，还是算法本身缺少很好的并行性造成的呢？下面通过计算Karp-Flatt量度[12]来回答这一问题。用e表示试验确定的串行比例，那么$e=\dfrac{1/S_p-1/p}{1-1/p}$。计算结果如表3所示。

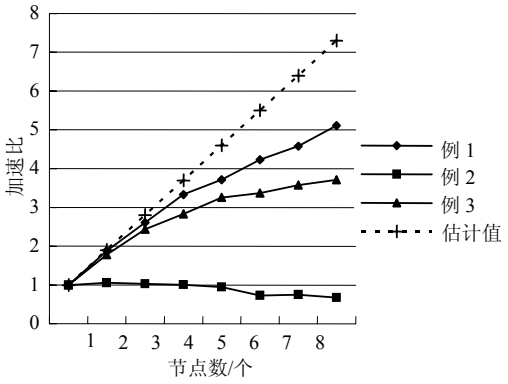

图7　并行前推法加速比曲线

表3　试验确定的串行比例

节点数	例 1	例 2	例 3
2	0.06	0.88	0.12
3	0.07	0.96	0.11
4	0.07	0.98	0.13
5	0.08	/	0.13
6	0.08	/	0.15
7	0.09	/	0.15
8	0.08	/	0.16

(1) 考察例1，e基本保持在一个稳定水平，表明性能不好的主要原因是算法中有限的并行性。

(2) 考察例2，当节点数大于5时，由于通信开销的急剧增大，使得e的计算失去了意义。

(3) 考察例3，发现e随着p增大而有规律地增大，说明此时算法和系统中的并行开销是导致加速比差的主要原因。

依据表2，计算并绘制数值并行算法的并行效率

曲线如图8所示。

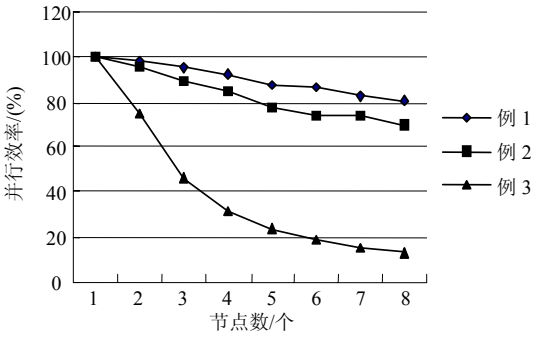

图8 数值并行算法效率曲线

从图8中可以看出，算法并行效率较高，并行效率随节点数增加呈下降趋势，这是通信开销的正常增大造成的。

有趣的是，并行前推法对例2的效果好，对例3的效果不好，而数值并行法正好相反。怎样把两种算法的长处进行互补，提高系统的综合解题能力，需要进一步地研究。

4 结 束 语

本文尝试将并行计算与几何定理机器证明相结合，所做的工作是一个良好的开端。两种并行算法在基于MPI-2的并行计算环境下，能很好地发挥并行计算的优势，有效缩短构造性几何命题机器证明的时间。其中，并行前推法由于其实际推理过程的复杂性[13]，算法效率波动较大，试验数据表明，其并行性能还有进一步提高的空间。

参 考 文 献

[1] WANG P. Parallel polynomial operations on SMPs: an overview[J]. Journal of Symbolic Computation, 2001, 11(1): 377-396.

[2] 张景中, 高小山, 周咸青. 基于前推法的几何信息搜索系统[J]. 计算机学报, 1996, 19(10): 721,723-727.
ZHANG Jing-zhong, GAO Xiao-shan, ZHOU Shan-ching. The geometry information search system by forward reasoning[J]. Chinese Journal of Computers, 1996, 19(10): 721,723-727.

[3] CHOU S C, GAO X S, ZHANG J Z. Automated production of traditional proofs for constructive theorems[C]//Proof of 8th IEEE Symposium on Logic in Computer Science. Washington D.C. : IEEE Computer Society Press, 1993.

[4] YANG L, ZHANG J Z. A prover for parallel numerical method verification to a class of constructive geometric theorems[J]. Journal of Guangzhou University (Natural Science Edition), 2002, 1(3): 30-35.

[5] 邓米克. 证明构造性几何定理的数值并行法[J]. 科学通报, 1988, 33(24): 13-16.
DENG Mi-ke. The parallel numerical method for constructive geometric theorems proving[J]. Chinese Science Bulletin, 1988, 33(24): 13-16.

[6] 都志辉. 高性能计算并行编程技术-MPI并行程序设计[M]. 北京: 清华大学出版社, 2001.
DU Zhi-hui. Parallel programming for high performance computing-parallel programming with MPI[M]. Beijing: Tsinghua University Press, 2001.

[7] MICHAEL J Q. parallel Programming in C with MPI and openMP [M]. New York: McGraw-Hill Press, 2003.

[8] WILLIAM G, EWING L, RAJEEV T. Using MPI-2: advanced features of the message-passing interface [M]. Boston: MIT Press , 1999.

[9] 潘 斌, 郭红霞. 几何定理并行验证算法研究[J]. 计算机工程, 2007, 33(1): 16-18, 21.
PAN Bin,GUO Hong-xia.Reasearch on parallel algorithm of numerical verification for geometry theorem proving[J]. Computer Engineering, 2007, 33(1): 16-18, 21.

[10] GREGORY V W. Practical parallel programming [M]. Boston: MIT Press, 1995.

[11] GUSTAFSON J L. Reevaluating Amdahl's law[J]. Communication of ACM, 1988, 31(5): 532-533.

[12] KARP, ALAN H, HORACE P F. Measuring parallel processor performance[J]. Communications of ACM, 1990, 33(5): 539-543.

[13] 李 涛, 张 波, 李传中. 基于前向推理的平面解析几何自动推理系统研究与实现[J]. 计算机应用, 2006, 26(7): 1715-1717.
LI Tao, ZHANG Bo, LI Chuan-zhong. Automatic reasoning system of plane analytic geometry based on forward reasoning[J]. Journal of Computer Applications, 2006, 26(7): 1715-1717.

编 辑 熊思亮