计算机科学2003Vol. 30№. 8

基于 LogP 简化模型的矩阵求逆并行算法研究

曾庆华　孙世新　陈天麒
（电子科技大学　成都610054）

The Parallel Algorithm of Computing Converse Matrix on the Simplified LogP Model

ZENG Qing-Hua　SUN Shi-Xin　CHEN Tian-Qi
(University of Electronic Science and Technology of China, Chengdu 610054)

Abstract　LogP is becoming a practical parallel computation model that meets the demanding of parallel computers and parallel algorithms. So it is important to re-design parallel algorithms on the LogP model. This paper studies the parallel algorithm of computing converse matrix on the simplified LogP model, and gets the simulating results.

Keywords　LogP model, Converse matrix, Parallel algorithms, PVM

1. 引言

随着并行处理系统的发展，对矩阵算法的并行处理也提出了新的要求。以往普遍使用的 PRAM 并行计算模型由于不太符合实际并行机的特点，已不能很好地适应现代并行处理的要求。而新兴的 LogP 模型则正在成为符合实际并行机和并行算法的要求的并行处理模型。已有的多数矩阵并行算法大都是建立在 PRAM 并行处理模型机上，由于 PRAM 模型认为远地数据访问延迟为零，而在现代并行处理机系统如 SMP、MPP、COW 上，非本地数据的访问延迟是影响系统性能的一个非常重要的因素，这就使得已有的 PRAM 并行算法在实际的并行处理机上的运行效率一般都较低。因而在 LogP 模型上重新设计相应的并行算法就成为了迫切需要。

LogP 模型支持短信息，但由于矩阵并行处理中的一次通信量通常为长消息，因而不能基于原有的 LogP 模型直接分析各种矩阵运算的并行算法，必须将它适当推广。文[1]即提出了一种推广方法，它指出处理机发送/接受一次长度为 N 字节的消息所需时间为：

$$T=T_{SR}+T_W\cdot N$$

其中 T_{SR}为处理机发送和接受时间之和，T_W 为单位字节传递时间。该推广方法用两个参数 T_{SR}、T_W 表示了原有 LogP 模型中 L、o、g 三个参数描述的网络特性，因而我们称之为一种简化的 LogP 模型，本文基于这种简化的 LogP 模型对矩阵求逆并行算法进行研究。

2. 矩阵并行算法的设计

一个算法具有它内在的有序性，即我们必须按照特定的步骤执行，才能够最终得到有用的结果。如果某些顺序错了，则会得到错误的结果。虽然并行算法允许顺序上某些操作可以并行，但由于数据结果的产生有其本身顺序上的要求，使得并行算法在总体上仍然是有序的。这种内在的有序性决定了现在的并行算法一般都是源于串行算法，大多数都是通过开发串行算法中的并行性而得到的，这也是本文中主要用到的方法。当然，串行算法不同，得到的并行算法也就会有所不同。

2.1　矩阵求逆的串行算法简介

矩阵求逆有多种方法，例如将矩阵 A 分解为 $A=LU$，其中 L 和 U 均为三角矩阵，可以用简便快速的方法求逆，从而得到 A 的逆矩阵；另外还可以令 $B=(A,E)$，对 B 进行初等变换，在将 A 变为单位矩阵 E 的同时，右边的单位矩阵 E 就变为了 A 的逆矩阵；再如从逆矩阵的定义式 $A\cdot A^{-1}=E$ 中，可以将矩阵的 A^{-1}的第 i 个列向量 x_i 看作是线性方程组 $Ax_i=e_i$ 的解向量，其中 e_i 是单位矩阵 E 的第 i 个列向量，从而可以用解线性方程组的方法求出 A^{-1}；此外还有利用求特征多项式的 Csank 算法来求矩阵的逆以及用迭代法等多种求逆矩阵的方法。

以上算法中，LU 分解法、解方程组法一般都要用到各种消元法，初等变换法实际上也是消元法，消元法涉及到选取主元(列主元、行主元及全主元)问题，选取的方法对结果及其精度有较大影响，而且不能处理矩阵不可逆的情况；Csank 算法又太过于复杂，迭代法的初值选取和循环控制问题不好解决，而且循环意味着计算量较大。故本文采用了另一种方法来求矩阵的逆，它是从 Givens 旋转变换中得到的。其原理如下：

已知用满足一定条件的某正交矩阵 Q 去右乘矩阵 A 可以得到一个上三角矩阵 R，即有 $Q\cdot A=R$，若 A 为可逆矩阵，则 R 的对角线元素均不为零；若 $rank(A)=r, r<N$，N 为矩阵 A 的阶，则 R 的前 r 个对角线元素不为零。于是当 A 可逆时可以得到 $A^{-1}=R^{-1}\cdot Q$，从而由 Q 和 R 就可以求出 A^{-1}；当 A 不可逆时，只要 R^{-1}是 R 的一个广义逆，那么可以证明，得到的 A^{-1}也是 A 的一个广义逆。本文的矩阵求逆并行算法就是将上述算法并行化而得到的。我们称之为 QR 法。

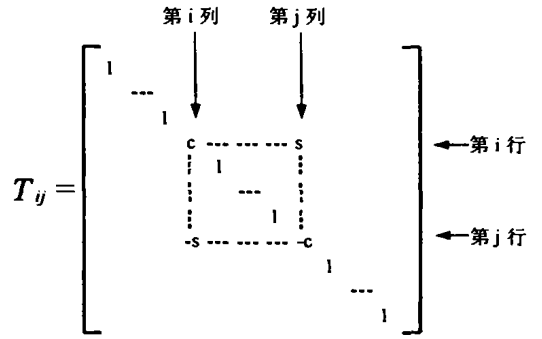
2.2　矩阵求逆的并行算法的设计

这里用 QR 法来求矩阵的逆，可以分三步来考虑其并行算法。(1)计算 $QA=R$ 时；(2)求 R^{-1}时；(3)计算 $R^{-1}Q$ 时。下面分别讨论。

第一部分，用 Q 乘以 A 求 R，实际上是对矩阵 A 进行 Givens 变换而将其化成一个上三角矩阵的过程。Givens 变换

曾庆华　博士，主要研究领域为阵列信号处理、多参量估计的并行算法研究。孙世新　教授，博士生导师，主要研究方向为计算机科学理论、并行算法、组合设计与组合优化等领域。陈天麒　教授，博士生导师，现在研究兴趣有阵列信号处理、自适应信号处理、现代谱估计、多传感数据融合、多参量估计的并行算法研究等。

的原理如下所述。设 T_{ij} 为某阶数合适的方阵，具有如下形式：



其中满足 $c^2+s^2=1$，称 T_{ij} 为初等旋转矩阵或者 Givens 旋转矩阵，显然 T_{ij} 是正交矩阵，而且行列式值为1。T_{ij} 具有一个很好的性质，就是当用它去左乘一个向量时，通过适当选取 c 和 s 的值，可以把该向量的第 j 个元素变为零，第 i 个元素则变为正数。例如设向量 $X=(x_1,x_2,\cdots,x_n)^T$，令 $Y=T_{ij}X$，$Y=(y_1,y_2,\cdots,y_n)^T$，于是有：

$$y_i=cx_i+sx_j,\ y_j=-sx_i+cx_j,$$

$$y_k=x_k,\ (k\neq i,j).$$

在 x_i 和 x_j 中，只要有一个不为零，那么若取

$$c=\frac{x_i}{\sqrt{x_i^2+x_j^2}},\ s=\frac{x_j}{\sqrt{x_i^2+x_j^2}}$$

则易验证 $y_i>0$，$y_j=0$。利用这个性质，我们可以用一系列的 T_{ij} 去左乘以矩阵 A，从而将矩阵 A 的下三角的元素全部消为零而得到上三角矩阵 R。这就是这一步变换的基本原理。容易看出每一个 T_{ij} 在左乘 A 矩阵时，与矩阵的乘法类似，存在着细粒度上的并行性；同时，由于每一个 T_{ij} 左乘 A 矩阵时，只改变 A 的第 i 行和第 j 行的元素，而其余行的元素保持不变，于是可以将若干个 T_{ij} 同时左乘以矩阵 A，使得它们分别改变 A 矩阵的不同的行，从而达到粗粒度上的并行。对于我们的系统来说，应尽量采用粗粒度的并行。

但是考虑到 T_{ij} 本身还需要用 A 矩阵的元素来计算，则要注意其中某些 T_{ij} 的计算有顺序上的限制，有一些 T_{ij} 必须在另一些 T_{ij} 的前面计算和去左乘 A 矩阵，否则将得到错误的结果。若 T_{ij} 表示消去 A 矩阵在位置 (i,j) $(i>j)$ 处的元素的变换，则对于 i 相同的 T_{ij}，其中 j 较小的变换必须先执行。根据以上规律和限制，Gentleman 提出了如下的并行处理方案：记集合 S 为 $S=\{(i,j):1\leqslant j<i\leqslant n\}$，$S$ 中的数对 (i,j) 对应着作用于第 $i-1$，i 两行而消去 A 矩阵的第 i 行 j 列的元素的变换 T_{ij}，于是 S 将按次序分解为 $S=\bigcup_{r=1}^{2n-3}S_r$，其中：

$$S_r=\{(i,j):1\leqslant j<i\leqslant n,\ n+2j=i+r+1\}$$

则子集 S_r 中的各数对所对应的变换彼此作用于不同的行，故可以并行执行，而 r 从1到 $2n-3$ 的顺序也保证了所有变换中对顺序的要求。易验证知 $S_r(r=1,2,\cdots,2n-3)$ 的元素个数依次为 $1,1,2,2,\cdots,n/2-1,n/2-1,n/2,n/2-1,n/2-1,\cdots,2,2,1,1$。还有一种对 S 的划分方法，对应另一种并行方法，基本思想是在每一步消去 A 矩阵同一列中的尽可能多的元素，所以称为贪婪算法，其所需要的总并行步数较 Gentleman 的方法为少，但其程序设计比较难以实现，故我们这里采用的是 Gentleman 的并行方法。

第二部分，求 R^{-1}，用初等变换法直接计算。由上可知，当 A 可逆时，R 的对角线元素均不为零，故首先将 R 的各行都除以对角线上的元素，再用第 i 行的元素消去第1～$i-1$ 行的元素，从而将 R 化为单位矩阵；而同样的变换则将单位矩阵化为了 R 的逆矩阵。其并行性依然有细粒度和粗粒度两种，细粒度并行性在于每消去某一行的元素时，同一行的元素可以并行执行；但我们这里仍然只考虑粗粒度上的并行性，在用第 i 行的元素消去第1～$i-1$ 行的元素时，这 $i-1$ 行的元素之间互不相关，可以并行执行。

第三部分，计算 $R^{-1}Q$，这里的 Q 与第一部分的 Q 相同，仍然是由一系列 T_{ij} 的乘积组成，同样可以知道，用 T_{ij} 去右乘 R^{-1} 只改变它的第 i 和 j 列而不影响其他的列。由于此时各个 T_{ij} 都已在第一部分中计算出来了，因此虽然在这里也可以采用与第一部分相同的并行方法，只是需要逆序进行，但是我们在这里还可以采用另一种并行度更高的方法。按照第一部分计算出来的 T_{ij} 的逆序，找出最前面的 $n/2$ 个不相交的 T_{ij}（不相交是指用它们去右乘矩阵 R^{-1} 时，只影响其不同的列），并行执行这部分计算；然后找下面 $n/2$ 个不相交的 T_{ij} 来进行下一步并行计算，直至计算完毕。其平均并行度显然要比第一部分的并行方法的平均并行度高得多。

3. 矩阵并行算法的分析

并行算法的实现过程实际上是一个映射过程。并行算法在设计时所注重的是最大限度地开发已有的串行算法在粗、中、细各种粒度上的并行性，这时可以认为它受并行体系结构的影响不大；而一个设计好的并行算法的性能好坏、效率高低，只有把它在某个有具体并行体系结构的并行机上实现以后，才能够对它进行评价，因为与性能、效率密切相关的计算调度、通讯调度只有在具体的并行机上进行才有意义。并行算法的实现过程就是把设计好的并行算法中的并行任务映射到并行机的相应结点上的过程，在这个过程中，需要着重考虑的有两点：如何最大限度地利用并行机来发挥出已设计好的并行算法的并行作用问题（计算调度）以及为尽量减少数据通讯和交换而需要考虑的原始数据分配问题以及通讯方法问题（通讯调度）。

3.1 矩阵求逆算法的计算复杂性

从算法的设计中可以知道，整个计算包括以下几个部分：计算 QA 和 $R^{-1}Q$ 部分；求 R^{-1} 部分；求矩阵 Q 的部分。下面分别进行分析：

第一部分，计算 QA 和 $R^{-1}Q$ 时，其计算过程类似，故放在一起考虑。矩阵 Q 由 $N(N-1)/2$ 个（A 的下三角元素的个数）Givens 变换矩阵乘积而得，每个 Givens 矩阵与 A 或者 R^{-1} 相乘时，都需要 $4N$ 次乘法和 $2N$ 次加法总共 $6N$ 次运算，所以得这时的总算术运算量为：

$$\frac{N(N-1)}{2}\cdot 6N+\frac{N(N-1)}{2}\cdot 6N=6N^3-6N^2$$

第二部分，求 R^{-1} 时，用初等变换法，它是依次用第 i 行（$i=2,3,\cdots,N$）的元素消去第 $i-1$ 行的第 i 至第 N 个元素。首先各行要除以对应对角线上的元素，共 $N(N-1)/2$ 个除法；在用第 i 行的元素消去第1～$i-1$ 行的元素时，每消去一行，需要 $N-i+1$ 次乘法和 $N-i+1$ 次加法共 $2(N-i+1)$ 次运算，故消去时的算术运算总量为：

$$\sum_{i=2}^{N}\sum_{j=1}^{i-1}2(N-i+1)=\sum_{i=1}^{N-1}(2Ni-2i^2)$$

$$=N^2(N-1)-N(N-1)(2N-1)/3$$

所以这一部分的总算术运算量为：

$$\frac{N(N-1)}{2}+N^2(N-1)-\frac{N(N-1)(2N-1)}{3}$$

$$=\frac{1}{3}N^3+\frac{1}{2}N^2-\frac{5}{6}N$$

第三部分，计算矩阵 Q 时，Q 由 $N(N-1)/2$ 个 Givens 变

（下转第184页）

```
        WHEN LDRR => CON<="0010000"&DR&"…;
        …
WHEN S6 => CASE OP IS
    WHEN CALA => CON<="10000…";--
WHEN S7 => CASE OP IS
    WHEN CALA => CON<="000010…;
WHEN OTHERS=>NULL;
END CASE;
```

小结 用 VHDL 语言设计复杂电路时，从大量的数据中能够分析出控制的重要方法是：电路中算术运算多，还是控制（多路选择）多，后者的优化和综合要求识别在转移内部和转移之间布尔简化的可能。上例中，CPU 的指令系统的每条指令均在一个状态时，产生所需控制信号，而每条指令所经历的状态是不同的。上述 P-T 算法模型是解决这类问题的一个较为简洁的好方法。

参 考 文 献

1 林敏. VHDL 数字系统设计与高层次综合第1版. 方颖立等，北京：电子工业出版社，2002. 1

2 卢毅. VHDL 与数字电路设计第1版. 赖杰，北京：科学出版社，2001. 4

3 王诚. 计算机组成与设计第1版. 北京：清华大学出版社，2002. 8

4 王诚. 计算机组成与设计实验指导第1版. 刘卫东等，北京：清华大学出版社，2002. 7

（上接第177页）

换组成，每一个 Givens 变换的获得需要5次算术运算和一次平方根运算，因此总共有 $\frac{5}{2}N^2-\frac{5}{2}N$ 次算术运算和 $\frac{1}{2}N^2-\frac{1}{2}N$ 次平方根运算。

综上可知，把三部分的计算量相加得总计算量为：$\frac{19}{3}N^3-3N^2-\frac{10}{3}N$ 次算术运算和 $\frac{1}{2}N^2-\frac{1}{2}N$ 次平方根运算。

3.2 并行度

并行度的计算也要分为三个部分：Q 乘以 A 处，求 R^{-1} 处，以及 R^{-1} 乘以 Q 处。在 Q 乘以 A 处，这一步用 $2N-3$ 个并行步完成，从算法的设计过程中易知每一步的并行度分别为 $1,1,2,2,\cdots,N/2-1,N/2-1,N/2,N/2-1,N/2-1,\cdots,2,2,1,1$；在求 R^{-1} 处，需要 $N-1$ 个并行步完成，由其算法的设计易知并行度依次为 $1,2,\cdots,N-1$；在 R^{-1} 乘以 Q 处，其算法设计的原理与 Q 乘以 A 时类似，也用 $2N-3$ 个并行步完成，每步的并行度也分别为 $1,1,2,2,\cdots,N/2-1,N/2-1,N/2,N/2-1,N/2-1,\cdots,2,2,1,1$。于是容易算得平均并行度为：$(3N^2-3N)/(10N-14)$。

3.3 通讯复杂度

按照前文所述的并行求逆算法，在假设有充分多的处理机的条件下来分析其通讯复杂度。同样分三个部分来分析：第一部分，计算 QA 时，共在 $2N-3$ 个并行步内完成，设第 i 个并行步内的并行度为 D_i，则在每个并行步中有 $D_i\cdot 2\cdot 8N$ 个字节的数据需要通讯，故这一步中的通讯复杂度为 $\sum_{i=1}^{2N-3}(D_i\cdot T_{SR}+T_W\cdot D_i\cdot 2\cdot 8N\cdot d)$。其中 T_{SR} 为发送处理机的发送开销和接收处理机的接收开销之和，T_W 为网络带宽的倒数，即每字节传输时间，d 为处理机间的平均通讯距离；第三部分，计算 $R^{-1}Q$ 时，采用同第一部分相同的并行策略，通讯复杂度也相同；第二部分，计算 R^{-1} 时，共在 $N-1$ 个并行步内完成，第 i 步的并行度为 i，每个并行步中共有 i 行 $i\cdot 8N$ 个字节的数据需要通讯，故这一步中的通讯复杂度为 $\sum_{i=1}^{N-1}(i\cdot T_{SR}+T_W\cdot i\cdot 8N\cdot d)$。综上可得总通讯复杂度为：

$$T=2\cdot\sum_{i=1}^{2N-3}(D_i\cdot T_{SR}+T_W\cdot D_i\cdot 2\cdot 8N\cdot d)+\sum_{i=1}^{N-1}(i\cdot T_{SR}+T_W\cdot i\cdot 8N\cdot d)$$

3.4 效率及加速比

若记以上算出的矩阵求逆算法的总计算复杂度为 W_2，总通讯复杂度为 T_2，则其算法的效率可通过式 $E_2=W_2/(W_2+T_2)$ 计算出来。而加速比则为 $\alpha_2=P\cdot E_2$，其中 P 为处理机的台数。

4. 模拟结果及分析

模拟环境：Intel pentium166MMX 软件平台：SCO Unix OpenServer 5. 20 PVM 3. 3110M 局域网。

有关矩阵求逆的模拟计算时间情况见表1。

表1 矩阵求逆模拟程序计算时间比较

矩阵阶数	串行时间(秒)	模拟4节点并行		模拟8节点并行		模拟16节点并行	
		时间(秒)	效率	时间(秒)	效率	时间(秒)	效率
64	1	15	27%	35	23%	70	23%
128	2	31	27%	68	24%	138	23%
256	10	68	59%	139	58%	278	58%

从表中计算时间的发展规律来看，当矩阵阶数增加一倍时，串行计算时间增加接近10倍，而并行计算时间只增加近两倍。因此可以预见的是，当矩阵阶数从256进一步增大，即计算量进一步增大，并行效率还应当能有适当提高。

结束语 本文提出一种基于 LogP 简化模型的矩阵求逆并行算法，该算法被用在 L 阵实现信号的频率、DOA 和极化的联合估计的并行处理中收到了良好的效果。实践验证了本算法的有效性。

参 考 文 献

1 李晓梅，等. 可扩展的并行算法设计与分析. 国防工业出版社，1999

2 Dowling E M，Fu Z，Drafz R S. HARP：An Open Architecture for Parallel Matrix and Signal Process. IEEE Transactions on Parallel and Distributed Systems，Oct. 1993

3 Tsay J-C，Chang P-Y. Design of Efficient Regular Arrays for Matrix Multiplication by Two-Step Regularization. IEEE Transactions on Parallel and Distributed Systems，Feb. 1995

4 Hwang K，Xu Z W. Scalable Parallel Computers for Real-Time Signal Processing. IEEE Signal Processing Magazine，1996

5 李晓梅，蒋增荣. 并行算法. 湖南科学技术出版社，1992

6 陈国良. 并行算法的设计与分析. 中国科学技术出版社，1993

7 李晓梅，等. 面向结构的并行算法——设计与分析. 国防科技大学出版社，1996

8 张林波，迟学斌，等. 网络并行计算与分布式编程环境. 科学出版社，1996