# 三对角方程组的分布式 SPP 算法*

王正华　车永刚　赵文涛

(国防科技大学并行与分布处理国家重点实验室　长沙　410073)

摘　要　发展了单向并行分裂法 (SPP) 用于求解三对角和块三对角线性方程组, 算法考虑了三对角线性系统求解中文件I/O 及结果传输通信所占时间比例较大的特点, 充分利用了计算、文件I/O 与通信三者之间的重叠。分析了 SPP 算法的计算与通信开销。在工作站机群上进行了测试分析, 结果表明 SPP 算法适合于分布式计算。

关键词　三对角方程组, 并行算法, 加速比

分类号　TP301

## Distributed SPP Algorithm for Tridiagonal Equations

Wang Zhenghua　Che Yonggang　Zhao Wentao

(National Lab. for Parallel and Distributed Computing, NUDT, Changsha, 410073)

**Abstract**　SPP algorithm is developed for the solution of tridiagonal and block tridiagonal equations. In the solution of the tridiagonal system, file I/O and result transfers are time consuming. SPP algorithm has taken the characteristic into account. The overlapping of computation, file I/O and communication is fully exploited. The computation count and communication count are analysed. The algorithm is tested on NOWs. It shows that SPP algorithm is suitable for distributed computing.

**Key words**　tridiagonal equations, parallel algorithm, speedup

在科学与工程计算中经常遇到三对角型和块三对角方程组的求解问题, 例如, 计算流体力学中用有限差分或有限体积方法离散三维 N-S 方程, 经过近似因子分裂处理后, 可归结为三个块三对角方程组的求解, 再经过标量化处理后只要解五个三对角方程组[1]。由于在理论与实际应用中的重要性, 三对角或块三对角线性方程组的并行求解是数值并行算法研究的重要课题之一。

现在已有一些三对角方程组的并行求解法, 如循环约化法[2]、Wang 提出的分裂法[3]、P. H. Michielse 和 H. Van der Vorst 提出的 Michielse&Vorst 算法[4]等。但在分布式环境下, 有些算法如循环约化法的通信开销太大, 分裂法和 Michielse&Vorst 算法也没有有效减少计算量与通信量。

迟利华提出了双向并行分裂 (DPP) 法[5], 其通信建立次数为 $2(p-1)$, 通信量为 $3(p-1)$, 而且改变了消元次序, 减少了等待, 比 Wang 的分裂法和 Pichielse&Vors 算法有较大改进。但是其算法没有考虑数据文件的 I/O 及结果数据传输时的通信, 而这些对整个求解过程的时间开销影响较大。本文考虑了这些因素, 设计了单向并行分裂法 (SPP), 对三对角和块三对角方程组进行了求解与测试分析。

## 1　SPP 算法

### 1.1　SPP 算法解三对角方程组

设所求解的三对角方程组 $Ax = b$ 表示如下:

*　国家 863 项目资助
1999 年 5 月 13 日收稿
第一作者: 王正华, 男, 1962 年生, 副教授

$$\begin{bmatrix} d_1 & c_1 & & & & & \\ a_2 & d_2 & c_2 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & a_{n-1} & d_{n-1} & c_{n-1} & \\ & & & & a_n & d_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \\ b_{n-1} \\ b_n \end{bmatrix}$$

设处理机数为 $p$, 不失一般性, 设 $n = pk$, 将处理机编号为 $N_1$, $N_2$, …, $N_p$。系数矩阵 $A$ 和右端项 $b$ 被预先按顺序平均分配到各处理机上。消元过程中引入的新的非零元就存储在 $a_i$ 和 $c_i$ 中, $i = 1, \cdots, n$。算法描述如下:

步骤一: 各处理机读取各自的数据。

步骤二: 对 $N_1$, 首先将第一行对角元化为 1, 然后自上往下用第 $i - 1$ 行消去本机上的下次对角元 $a_i$, $2 \le i \le k$, 消去每一行的下次对角元后, 将该行的对角元化为 1。对 $N_j$, $2 \le j \le p - 1$, 首先自上往下用第 $i - 1$ 行消去第 $i$ 行的下次对角元 $a_i$, $(j - 1)k + 2 \le i \le jk$, 消去每一行的下次对角元之后, 将该行的对角元化为 1, 再自下往上用第 $i + 1$ 行消去第 $i$ 行的上次对角元 $c_i$, $(j - 1)k + 1 \le i \le jk - 2$。对 $N_p$, 首先将最后一行的对角元化为 1, 然后自下往上用第 $i + 1$ 行消去第 $i$ 行的上次对角元 $d_i$, $n - k + 1 \le i \le n - 1$, 消去每一行的上次对角元之后将该行的对角元化为 1。经过步骤二后, 各处理机上的系数矩阵形状如下(图中略去了一些元素全为零的块, 空白处为零元素, 下同):

$$\begin{bmatrix} 1 & * & & & & \\ & 1 & * & & & \\ & & \ddots & \ddots & & \\ & & & 1 & * & \\ & & & & 1 & * \end{bmatrix} \quad \begin{bmatrix} * & & 1 & & & * \\ * & & & 1 & & * \\ & & & & \ddots & \\ * & & & & 1 & * \\ * & & & & & 1 & * \end{bmatrix} \quad \begin{bmatrix} * & & 1 & & & \\ & * & & 1 & & \\ & & \ddots & \ddots & & \\ & & & * & & 1 \\ & & & & * & 1 \end{bmatrix}$$

$N_1$ $N_j(2 \le j \le p - 1)$ $N_p$

步骤三: 对 $N_p$, 将本机上的第一行数据中的 $a_{(p-1)k+1}$, $b_{(p-1)k+1}$ 向上传给 $N_{p-1}$(对角元 $d_{(p-1)k+1}$ 已经化为 1, 不必传, 下同)。对 $N_j$, $2 \le j \le p - 1$, 首先接收来自 $N_{j+1}$ 的数据, 用接收来的数据消去本机上最后一行数据中的 $c_{jk}$, 再将最后一行的对角元化为 1, 然后用本机最后一行消去本机第一行数据中的 $c_{(j-1)k+1}$, 再将本机第一行数据中的 $a_{(j-1)k+1}$、$b_{(j-1)k+1}$ 向上传给 $P_{j-1}$, 最后用本机最后一行消去本机其余各行数据中的 $c_i$, $(j - 1)k + 2 \le i \le jk - 1$。对 $N_1$, 首先接收 $N_2$ 传来的数据, 并用接收来的数据消去本机上最后一行数据中的 $c_k$, 再将最后一行对角元化为 1。经过步骤三后, 各处理机上的系数矩阵形状如下:

$$\begin{bmatrix} 1 & * & & & & \\ & 1 & * & & & \\ & & \ddots & \ddots & & \\ & & & 1 & * & \\ & & & & 1 & * \end{bmatrix} \quad \begin{bmatrix} * & & 1 & & & \\ * & & & 1 & & \\ * & & & & \ddots & \\ * & & & & & 1 \\ * & & & & & & 0 \end{bmatrix} \quad \begin{bmatrix} * & & 1 & & & \\ & * & & 1 & & \\ & & \ddots & \ddots & & \\ & & & * & & 1 \\ & & & & * & 1 \end{bmatrix}$$

$\boldsymbol{N_1}$ $\boldsymbol{N_j(2 \le j \le p - 1)}$ $\boldsymbol{N_p}$

步骤四: 对 $N_1$, 首先将本机上最后一行数据中的 $b_k$ 向下传送给 $N_2$, 然后自下向上用第 $i + 1$ 行消去第 $i$ 行的上对角元 $d_i$, $1 \le i \le k - 1$。对 $N_j$, $2 \le j \le p - 1$, 首先接收 $N_{j-1}$ 传来的数据, 用接收来的数据消去本机最后一行数据中的 $a_{jk}$, 再将本机最后一行数据中的 $b_{jk}$ 向下传给 $N_{j+1}$, 最后用接收来的数据消去本机其余各行数据中的 $a_i$, $(j - 1)k + 1 \le i \le jk - 1$。对 $N_p$, 首先接收 $N_{p-1}$ 传来的数据, 用接收来的数据消去本机上第一行数据中的 $a_{n-k+1}$, 然后自下向上用第 $i - 1$ 行消去第 $i$ 行数据中的 $a_i$, $n - k + 2 \le i \le n$。经过步骤四后, 各机上的右端项 $b_i$ $(1 \le i \le n)$ 的值就是方程组的解。

步骤五: 对 $N_1$, 首先将本机上的结果写入文件, 然后依次接收 $N_2 \sim N_p$ 传送来的数据, 并将其写入文件。对 $N_2 \sim N_p$, 将本机上的数据传送给 $N_1$。

**1.2** SPP 算法解块三对角方程组

将三对角方程组系数矩阵中的元素换成等阶的方阵，右端项的元素换为等长度的向量，就成为块三对角方程组。块三对角方程组的解法跟三对角方程组的解法相似，不同之处在于前者中的对单个元素的操作在后者中变成对矩阵块或向量的操作，另外在后者中邻机间每次传送的数据不是一行而是几行。

**1.3** 算法说明

数据平均分配给各台处理机，各机的负载基本相同，而且数据匀齐，便于程序处理。

根据系数矩阵的特点，$N_1$ 和 $N_p$ 在步骤二只进行单边消元，跟 DPP 算法相比减少了计算量(对三对角方程组求解而言，减少了 $4n/p$ 个乘法和 $4n/p$ 个减法)。

使用单向并行分裂，并且让数据传送的方向是先由后向前传，后由前向后传，使得编号越小的处理机越早完成计算，$N_1$ 按编号顺序接收各个处理机传来的结果，减少了后面的处理机的等待时间，而且利用了计算、文件 I/O 与通信的重叠。

步骤二中，$N_2 \sim N_{p-1}$ 的向下和向上消元并未使系数矩阵中的非零元减少，但它使得在步骤三、步骤四中可以不是逐行消元而是跳过中间各行先消去本机上第一行或最后一行的非对角的非零元，之后立即将消元后的一行数据传给邻机，减少了各处理机之间的等待时间，并且也利用了计算与通信的重叠。

## 2 测试结果及分析

**2.1** 测试结果

在通过 10 兆以太网互联的 NOWs 上进行测试，结点机为 Pentium 166，内存各 64MB。串行计算使用已知的最快算法追赶法，块三对角求解中块大小为 3×3，块求逆使用高斯消去法。测试结果如表 1、表 2，其中 $p$ 为处理机数，$n$ 为方程组的阶数，$R$ 为 $N_1$ 读数据的时间占其整个运行时间的百分比，$Sp$ 为加速比，这里使用绝对加速比，其计算式为

$$Sp = \frac{\text{最快的串行算法在单机上的执行时间}}{\text{并行算法使用 } m \text{ 台处理机的执行时间}}$$

表 1 对三对角方程组的测试结果

Tab.1 The Performance for tridiagonal equations

| $n$ | 10000 | | | 40000 | | | 160000 | | | 640000 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p$ | 2 | 4 | 6 | 2 | 4 | 6 | 2 | 4 | 6 | 2 | 4 | 6 |
| $R$ | 60% | 48% | 30% | 62% | 44% | 32% | 61% | 44% | 34% | 61% | 43% | 32% |
| $Sp$ | 1.65 | 2.38 | 2.59 | 1.56 | 2.09 | 2.22 | 1.59 | 2.33 | 2.61 | 1.61 | 2.16 | 2.54 |

表 2 对块三对角方程组的测试结果

Tab.2 The performance for block tridiagonal equations

| $n$ | 3600 | | | 14400 | | | 57600 | | | 230400 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p$ | 2 | 4 | 6 | 2 | 4 | 6 | 2 | 4 | 6 | 2 | 4 | 6 |
| $R$ | 64% | 49% | 39% | 70% | 48% | 34% | 72% | 54% | 44% | 72% | 55% | 45% |
| $Sp$ | 1.88 | 2.79 | 3.13 | 2.02 | 2.74 | 2.89 | 2.06 | 3.07 | 3.78 | 2.01 | 3.08 | 3.80 |

**2.2** 结果分析

SPP 算法所需的存储量和 I/O 量跟串行追赶法的一样。计算开销的主要部分是乘除运算，三对角串、并行算法乘除法的计算开销分别为 $5n$ 和 $(12p - 14)n/p$，块三对角串、并行算法的乘除法的计算开销分别为 $34n$ 和 $(99p - 130)n/p$。

SPP 求解三对角方程组的通信量为 $3(p - 1) + (p - 1)(n/p)$，求解块三对角方程组的通信量为

$15(p-1)+(p-1)(n/p)$, 二者通信建立次数均为 $3(p-1)$, 与 DPP 的相同。

令 $s$ 为并行求解的计算量与串行计算量之比值, $p$ 为处理机台数, 则 $p/s$ 为不考虑I/O及通信开销的理论加速比。分别以三对角和块三对角求解时的一个算例为例, 实际运行时的加速比 $Sp$ 与理论预测值的对比如表3、表4。

表3 三对角SPP算法($n=160000$)的理论加速比与实测加速比

Tab.3 Theoretical and real speedup of SPP algorithm for tridiagonal equations

| $p$ | 2 | 4 | 6 | $\infty$ |
|---|---|---|---|---|
| $s$ | 1 | 1.70 | 1.93 | 2.40 |
| $p/s$ | 2 | 2.35 | 3.11 | $0.42P$ |
| $Sp$ | 1.59 | 2.33 | 2.61 | —— |

表4 块三对角SPP算法($n=57600$)的理论加速比与实测加速比

Tab.4 Theoretical and real speedup of SPP algorithm for block tridiagonal equations

| $p$ | 2 | 4 | 6 | $\infty$ |
|---|---|---|---|---|
| $s$ | 1 | 1.96 | 2.27 | 2.91 |
| $p/s$ | 2 | 2.04 | 2.64 | $0.34P$ |
| $Sp$ | 2.06 | 3.07 | 3.78 | —— |

当处理机数为2时, 实测加速比较为理想, 甚至出现了超线性。这主要是因为使用2台处理机时计算量跟串行时一样, 且由于cache的影响, 使得访存的平均时间缩短, 因而获得了超线性加速比。

处理机数为4和6时, 加速比不太理想, 是因为处理机数大于2时总要因并行而引入冗余计算, 且随p增大冗余计算量也增大。但实测加速比都比理论加速比大, 是因为三对角型方程组求解, I/O占整个运行时间的比例很大, 而开始的文件I/O部分是完全并行的, 对加速比贡献较大。此外, cache对实测加速比的提高也略有贡献。

比较表2和表1可看出, 算法对块三对角方程组求解的加速效果比对三对角方程组加速效果好得多, 这主要是因为二者的通信模式一样, 启动通信的次数相同, 前者只在计算时要比后者多传数据, 收集结果时传送的数据量跟后者相同, 通信开销比后者大不了多少, 而其计算量是后者的几倍, 相对而言, 它的通信时间占总时间的比例要小得多。另外, 前者中开始读数据的时间所占比例比后者中的大, 而这一阶段是完全并行的。综合这些原因, 前者的并行效果要好于后者。

## 3 结论

对三对角和块三对角方程组的并行分裂求解而言, 处理机数大于2时总要引入冗余计算, 所以处理机数大于2时的加速比理论上就不能很高。但是由表3、4可以看出, 随着处理机数的增加, 并行求解的计算量与串行计算量的比值逐步趋于一个固定的常数, 处理机数目达到一定规模后, 增大处理机数对这一比值的影响就很不明显了。相反, 处理机数目增大时, 文件I/O处理部分并行度更高, 核心计算部分并行度更高, 所以实测加速比会随处理机数的增加而增加。本文所发展起来SPP算法不失为分布式环境下求解三对角和块三对角方程组的一种较为理想的算法。

## 参考文献

1 贺国宏. 三阶ENN格式及其在高超声速粘性复杂流场求解中的应用[博士学位论文]. 中国空气动力研究与发展中心, 1994: 38～48

2 Amodio P et al. A parallel version of the cyclic reduction algorithm on a hypercube. Parallel Computing, 1993, 19: 1273～1281

3 Wang H H. A parallel method of tridiagonal equations. ACM Trans. on Math software, 1981, 7 (2): 170～183

4 Michielse P H, Van Vorst H A. Data tranSport in Wang's partition method, Parallel Computing, 1988, 7: 87～95

5 迟利华. 大型稀疏线性方程组在分布式存储环境下的计算[博士学位论文]. 国防科技大学计算机系, 1998: 22～26