

# 按行及按列划分的并行矩阵向量乘法的分析

黎凯伦, 吴伟民, 苏 庆

(广东工业大学 计算机学院, 广东 广州 510090)

**摘 要:**文中首先总结按行划分和按列划分的并行矩阵向量乘法在原理上的异同。然后实现基于 MPI 模型的按行划分以及按列划分的矩阵向量乘法的程序,并分析了程序在基本框架方面的异同。最后给出测试这两种程序的实验条件和任务,并对这两种程序在不同情况下的执行时间进行分析。

**关键词:**并行计算;矩阵向量乘法;按行划分;按列划分;MPI

**中图分类号:**TP338.6

**文献标识码:**A

**文章编号:**1005-3751(2006)04-0041-03

## Analysis on Parallel Matrix - Vector Multiplication Based on Divided by Row and Divided by Column

LI Kai-lun, WU Wei-min, SU Qing

(Computer Faculty, Guangdong University of Technology, Guangzhou 510090, China)

**Abstract:** First summarizes the differences on principle between two kinds of parallel algorithm of matrix - vector multiplication, namely, divided by row and divided by column. Based on these two algorithms, implements two programs accordingly using the MPI model and analyzes differences on the frame of these two programs. In the end, puts forward the condition and task of the experiment which purpose is to test these two programs and analyzes the executing time under diverse circumstances.

**Key words:** parallel computing; matrix - vector multiplication; divided by row; divided by column; MPI

### 1 并行计算技术

#### 1.1 并行计算

在当今的信息社会,计算机需要处理的信息量越来越大,需要解决的问题越来越复杂,使得计算量剧增。通过提高单个处理器的计算速度和采用传统的“串行”计算技术已难以胜任。因此,需要有功能更强大的计算机系统和高效计算技术来支撑。并行计算机及并行计算技术也就应运而生。并行计算就是在并行计算机或者分布式计算机等高性能计算系统上所作的超级计算。它和常说的高性能计算、超级计算是同义词,因为任何高性能计算和超级计算总离不开使用并行技术。它的物质基础是高性能并行计算机(包括分布式网络计算机)<sup>[1]</sup>。

#### 1.2 矩阵向量乘法概述

矩阵向量乘法在数值计算和非数值计算中扮演着十分重要的角色(例如线性方程组的求解、多项式的求解)。当矩阵的阶数很小的时候,单 CPU 的计算机很容易处理,但是当矩阵的阶数很大的时候,单 CPU 的处理机进行计算出现了困难,或者说成为了不可能。这时,必须将矩阵

向量乘法的算法并行化,采用多 CPU 的计算机(或者分布式网络计算机)运行该并行算法。这样既能降低时间复杂度,又能充分利用多 CPU 处理机的资源<sup>[2]</sup>。

### 2 矩阵向量乘法的算法原理

对于并行算法的实现,首先考虑的是如何划分。对于矩阵向量乘法,通常有两种划分方法:带状划分和棋盘划分,其中带状划分包括按行划分以及按列划分<sup>[3]</sup>。

#### 2.1 按行划分的算法原理

为了讨论的方便,在按行划分以及按列划分的算法原理讨论中,设定矩阵为方阵。至于矩阵为非方阵的情况,是方阵情况的简单扩展。

在按行划分的情况下,假设处理器数目  $p$  与矩阵的阶  $n$  相等。计算前处理器  $P_i$  存放向量的第  $i$  个分量  $x_i$  和矩阵的第  $i$  行  $a_{i,0}, a_{i,1}, \dots, a_{i,n-1}$ , 并将  $x_i$  向其余的处理器进行多到多的播送。播送完毕后处理器  $P_i$  进行  $\sum_{j=0}^{n-1} a_{ij}x_j$  的计算,并将计算结果赋予  $y_i$ 。计算完毕后,  $P_i$  将  $y_i$  播送给一个指定的处理器,该处理器将  $y_i$  进行组合得出结果向量  $y^{[1]}$ 。在这种情况下,通信时间为  $O(n)$ , 计算时间也为  $O(n)$ 。

当处理器数目  $p$  小于矩阵的阶  $n$  时,情况是类似的,只是开始计算前每个处理器存放矩阵的  $n/p$  行(元素个

收稿日期:2005-08-12

作者简介:黎凯伦(1981-),男,广东新会人,硕士研究生,研究方向为可视计算、多媒体网络;吴伟民,副教授,研究生导师,研究方向为数据结构与算法、可视计算、程序语言系统、嵌入式系统。

数为  $n^2/p$ ) 和向量的  $n/p$  个分量<sup>[1]</sup>。显然,通信时间为  $O(n/p)$ ,计算时间也为  $O(n^2/p)$ 。

## 2.2 按列划分的算法原理

按列划分与按行划分的异同点如下:

1) 与按行划分相比,计算前每个处理器存放矩阵和向量的  $n/p$  列,而不是  $n/p$  行。但是保存的元素总数是相同的,每个处理器  $P_i$  保存矩阵的  $n^2/p$  个元素和向量的  $n/p$  个分量<sup>[1]</sup>。

2) 按列划分的计算原理和按行划分的计算原理不同。按列划分采用的是外积的算法,  $AX = (a_{11}, a_{21}, \dots, a_{N1})^T \times 1 + (a_{12}, a_{22}, \dots, a_{N2})^T \times 2 + \dots + (a_{1N}, a_{2N}, \dots, a_{NN})^T \times N$ , 而按行划分采用的是内积的算法,  $AX = (\sum_{j=1}^N a_{1j}x_j, \sum_{j=1}^N a_{2j}x_j, \dots, \sum_{j=1}^N a_{Nj}x_j)^T$ <sup>[4]</sup>, 虽然计算原理不同,但是计算时间与按行划分相同,也是  $O(n^2/p)$ 。

3) 与按行划分相比,按列划分不用进行向量元素的多到多播送。但是,在组合结果时,由于采取循环部分和的方法将局部计算的结果累加,所以通信时间与按行划分相同,也是  $O(n/p)$ 。

## 3 矩阵向量乘法在 MPI 上的实现

### 3.1 MPI 概述

MPI 是目前应用最广的并行程序设计平台,几乎被所有并行计算环境(共享和分布式存储并行机、MPP 以及机群系统等)和流行的多进程操作系统(UNIX, Windows NT)所支持<sup>[5]</sup>。

### 3.2 矩阵向量乘法在 MPI 上的实现

#### 3.2.1 按行划分的算法实现

实现按行划分的矩阵向量乘法的程序的基本思想如下:

主进程首先将向量  $B$  广播给所有的从进程,然后将矩阵  $A$  的各行依次发送给从进程;从进程首先计算  $A$  的一行和  $B$  相乘的结果,然后将结果发送给主进程;主进程循环向各个从进程发送一行数据,直到将  $A$  各行的数据发送完毕;一旦主进程将  $A$  的各行发送完毕,则每收到一个结果,就向相应的从进程发送结束标志;从进程接收到结束标志后结束,主进程收集完所有的结果后也结束。

程序实现的关键点如下:

1) 主进程调用  $\text{MPI\_Bcast}(B, \text{COL} * 1, \text{MPI\_INTEGER}, 0, \text{MPI\_COMM\_WORLD})$  进行向量  $B$  的多到多播送;

2) 从进程调用  $\text{MPI\_Bcast}(t2, \text{COL} * 1, \text{MPI\_INTEGER}, 0, \text{MPI\_COMM\_WORLD})$  接收向量  $B$ ;

3) 主进程调用  $\text{MPI\_Send}(A[\text{row}], \text{COL}, \text{MPI\_INTEGER}, i, \text{row}, \text{MPI\_COMM\_WORLD})$  发送矩阵的第  $i$  行数据至第  $i$  个从进程;调用  $\text{MPI\_Recv}(\&t1, 1, \text{MPI\_INTEGER}, \text{MPI\_ANY\_SOURCE}, \text{MPI\_ANY\_TAG}, \text{MPI\_COMM\_WORLD}, \&\text{status})$  接收  $A$  的行向量与向量  $B$  相

乘后得出的结果;

4) 从进程调用  $\text{MPI\_Recv}(t1, \text{COL}, \text{MPI\_INTEGER}, 0, \text{MPI\_ANY\_TAG}, \text{MPI\_COMM\_WORLD}, \&\text{status})$  接收  $A$  的行向量;调用  $\text{MPI\_Send}(\&t3, 1, \text{MPI\_INTEGER}, 0, \text{row}, \text{MPI\_COMM\_WORLD})$  把保存  $A$  的行向量和向量  $B$  的内积结果的变量  $t2$  发送至主进程。

根据按行划分的程序,主进程需要广播  $\text{COL}$  个向量的分量。主进程发送  $\text{ROW}$  个包(每个包的大小是  $\text{COL}$ , 表示矩阵元素)给从进程,从从进程接收  $\text{ROW}$  个包(每个包的大小是 1, 表示计算结果)。从进程计算每一个结果向量的分量需要  $\text{COL}$  次加法和乘法,总共需要计算  $\text{COL} * \text{ROW}$  次加法和乘法。

#### 3.2.2 按列划分的算法实现

实现按列划分的矩阵向量乘法的程序的基本思想如下:

主进程首先将矩阵  $A$  的各列和对应的向量的分量依次发送给从进程;从进程计算一列和向量元素相乘的结果,然后将结果发送给主进程;主进程循环向各个从进程发送一列数据和对应的向量分量,直到将  $A$  各列的数据发送完毕;一旦主进程将  $A$  的各列发送完毕,则每收到一个结果,就向相应的从进程发送结束标志;从进程接收到结束标志后结束,主进程收集完所有的结果后也结束。

按列划分的基本框架与按行划分的大致相同,不同的是:

1) 由于主进程发送的数据是矩阵  $A$  的一列数据,而矩阵  $A$  初始化时是按照行地址优先存储的。所以在并行计算前,需要把矩阵转置,把  $A$  改成是按照列地址优先存储;

2) 把向量的分量和矩阵  $A$  的列向量组织为一个矩阵,以便一起发送给相应的处理器;

3) 主进程调用  $\text{MPI\_Send}(A[\text{col}], \text{ROW} + 1, \text{MPI\_INTEGER}, i, \text{col}, \text{MPI\_COMM\_WORLD})$  发送矩阵的第  $i$  列数据和向量的第  $i$  个分量至第  $i$  个从进程;调用  $\text{MPI\_Recv}(t, \text{ROW}, \text{MPI\_INTEGER}, \text{MPI\_ANY\_SOURCE}, \text{MPI\_ANY\_TAG}, \text{MPI\_COMM\_WORLD}, \&\text{status})$  接收列向量与向量分量相乘后得出的向量;

4) 从进程调用  $\text{MPI\_Recv}(t1, \text{ROW} + 1, \text{MPI\_INTEGER}, 0, \text{MPI\_ANY\_TAG}, \text{MPI\_COMM\_WORLD}, \&\text{status})$  接收  $A$  的列向量和向量  $B$  的对应分量,存储于  $t1$ ;调用  $\text{MPI\_Send}(t2, \text{ROW}, \text{MPI\_INTEGER}, 0, \text{col}, \text{MPI\_COMM\_WORLD})$  把保存  $A$  的列向量和向量  $B$  的对应分量的乘积结果的  $t2$  向量发送至主进程;根据按列划分的程序,主进程不需要广播向量。主进程发送  $\text{COL}$  个包(每个包的大小是  $\text{ROW} + 1$ , 表示矩阵和向量元素)给从进程,从从进程接收  $\text{COL}$  个包(每个包的大小是  $\text{ROW}$ ),主进程总共需要计算  $\text{COL} * \text{ROW}$  次加法,从进程总共需要计算  $\text{COL} * \text{ROW}$  次乘法。与按行划分相比,主进程和从进程计算的总乘法次数和总加法次数是相等的,

不同的是每次发送和接收数据的数量。

4 实验结果

4.1 实验条件和任务概述

根据第 3 节描述的程序,在建立的 MPI 并行环境上分别运行不同规模的按行划分以及按列划分的矩阵向量乘法。由于并行计算适合于数据量较大的计算任务,所以矩阵规模分别采取 50×50,100×100,200×200,300×300。实验分别使用 1 个结点,2 个结点,3 个结点,4 个结点(每个结点都是单 CPU,型号都为 PIV 1.5GHz,512kL2,1ML3,操作系统都为 Windows XP)来完成计算。

在实验结果分析时需要注意影响并行计算性能的因素:

1)网络性能。在测试过程中,并行计算的时间是不稳定的,有时时间会变得很大。原因是本次实验的网络环境是利用实验室的局域网。这种抢占式的带宽分配造成并行计算网络环境的不稳定性,同时也影响了网络并行计算的性能;

2)矩阵和向量采用随机值初始化,不同特性的矩阵和向量相乘对运行时间有一定影响;

3)MPI 的任务调度。类似于把 8 个进程分配给 2 个结点,不等于每个结点有 4 个进程,因为在 MPI 的程序中只指定了进程的数目,而把不同进程进行组合和映射的工作是在 MPI 的底层部分实现,对用户透明<sup>[5]</sup>。

基于以上 3 点因素,对每一种情况下并行计算时间的确定,采用多次运行取平均值的方法。

4.2 实验结果分析

按行划分的并行计算时间(以秒为单位)如表 1 所示(括号内的她字表示进程数)。

表 1 按行划分的并行计算时间

矩阵规模	1 个结点(4)	2 个结点(8)	3 个结点(12)	4 个结点(16)
50×50	0.0269687	0.0448418	0.179386	0.1001190
100×100	0.0304387	0.0564664	0.300662	0.0870834
200×200	0.0461752	0.0672561	0.205023	0.0716447
300×300	0.0748596	0.0900141	0.258670	0.0985663

按列划分的并行计算时间(以秒为单位)如表 2 所示(括号内的数字表示进程数)。

表 2 按列划分的并行计算时间

矩阵规模	1 个结点(4)	2 个结点(8)	3 个结点(12)	4 个结点(16)
50×50	0.0233023	0.0262456	0.0319666	0.0560001
100×100	0.0287801	0.0386619	0.0459031	0.0636264
200×200	0.0332701	0.0517920	0.0577157	0.0631534
300×300	0.0476856	0.0738858	0.0656160	0.0744067

对比按行划分和按列划分的计算结果,可以得出以下几点结论:

1)在同等条件下,按行划分的时间比按列划分的时间大。由于测试案例的行和列都相等,导致在按行划分和按列划分情况下进程之间的通信时间相差很少以及所有结点的总计算量是一样的,所以运行时间的主要差别是按行划分需要花费广播向量至其他的进程和其他的结点的时间,而按列划分不需要。

2)3 个结点下,按行划分的时间比按列划分的时间大一个数量级。因为在这种情况下,向量广播是按行划分计算的瓶颈,比多个任务并行计算的代价大的多。在这种条件下,最佳方案是按列划分。

5 结 论

文中笔者对按行划分以及按列划分的矩阵向量乘法在原理以及程序的执行时间方面的异同进行了分析。由于按列划分的算法在文献上涉及甚少,并且按行划分和按列划分在原理和执行时间上存在差异,所以在算法原理和程序上比较它们的异同,以便在不同条件下采用最佳的算法,具有一定的意义。

参考文献:

[1] 陈国良.并行计算-结构、算法、编程(修订版)[M].北京:高等教育出版社,2003.

[2] 陈国良.并行算法的设计与分析(修订版)[M].北京:高等教育出版社,2002.

[3] 张丽君,金绥更.向量算法与并行算法[M].北京:国防工业出版社,1993.

[4] 李晓梅,莫则尧,文尚猛.面向结构的并行算法-设计与分析[M].北京:国防科技大学出版社,1996.

[5] 都志辉.高性能计算之并行编程计算-MPI 并行程序设计[M].北京:清华大学出版社,1999.

[3] Gummadi K P, Dunn R J, Saroiu S, et al. Measurement, Modeling, and Analysis of a Peer-to-Peer File-sharing Workload[A]. Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP-19)[C]. Bolton Landing, NY, USA: [s. n.], 2003. 314-329.

[4] Fessant F L, Handurukande S, Kermarrec A M, et al. Clustering in Peer-to-Peer File Sharing Workloads[A]. Proceedings of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS)[C]. San Diego, USA: [s. n.], 2004.

[5] Sripanidkulchai K, Maggs B, Zhang H. Efficient Content Location using Interest-based Locality in Peer-to-Peer systems[A]. Proceedings of 22nd Annual Joint Conference of the IEEE Computer and Communications Societies[C]. San Francisco, CA, USA: [s. n.], 2003.

[6] Dabek F, Li J Y, Sit E J, et al. Designing a DHT for Low Latency and High Throughput[A]. Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI '04)[C]. Berkeley, CA, USA: [s. n.], 2003. 85-98.

(上接第 40 页)