第33卷 第24期 计 算 机 工 程 2007年12月
Vol.33 No.24 Computer Engineering December 2007
·软件技术与数据库· 文章编号：1000—3428(2007)24—0072—02 文献标识码：A 中图分类号：TP302.7

一种基于PVM的矩阵相乘并行算法

韦安定，李代平，文 剑
(广东工业大学计算机学院，广州 510006)

摘 要：研究了一种运行于PVM并行计算平台的矩阵相乘的并行算法。在工作站数量不为某个数的平方数时，Cannon算法在PVM环境下不能充分地利用机群系统中的资源。根据PVM并行编程环境中任务间通信的特点，文中设计了一种基于PVM的矩阵相乘并行算法，该算法根据工作站数量来确定子任务的数量，并对矩阵 ***A*** 进行分块，每个子任务可以计算一个分块。实验表明，该算法提高了机群并行环境中资源的利用率，提高了程序的运行效率。

关键词：矩阵相乘；并行算法；机群并行系统

Parallel Algorithm for Matrix Multiple Based on PVM

WEI An-ding, LI Dai-ping, WEN Jian
(College of Computer, Guandong University of Technology, Guangzhou 510006)

【Abstract】 This paper discusses a parallel algorithm for matrix multiple which run on the PVM. When the quantity of workstation is not for some integer square numbers, the Cannon algorithm can not fully use the resources. According to the features of task communication on the PVM, an effective parallel algorithm of matrix multiple is designed. Quantity of workstation determines that of task, in order to account a part for each task. Matrix ***A*** is compartmentalized. Experimental results show the algorithm improves resource utilization in fleet parallel environment, and improves operational efficiency of procedures.

【Key words】 matrix multiple; parallel algorithm; fleet parallel environment

网络并行计算环境是一种分布式处理系统，它利用高速网络互联的工作站机群软硬件资源，解决了“只能使用昂贵的大规模并行计算机系统才能完成的高性能计算”的问题。它同大型、超级并行计算机相比，具有投资小、灵活性强、应用软件开发速度快、可移植等优点[1]。网络并行计算环境成为当今并行处理技术研究的热点。PVM(Parallel Virtual Machine)[2]是一个免费的、公开源码的、被广泛接受的并行编程环境，提供资源管理、进程管理、消息传递以及协同操作等支持。它可以把多个异构的计算机组织起来成为一个并行计算系统。这些异构的计算结点可以通过多种网络互联成为一个网络计算虚拟机。

1 问题提出

矩阵乘法是一种被广泛应用的一种数值算法，在科学和工程计算的许多问题中涉及矩阵相乘运算。很多学者正在寻找实现矩阵相乘的高效算法。矩阵相乘的并行算法是解决大规模矩阵相乘的一种有效途径，在现有的矩阵相乘算法中，比较有影响的是Cannon算法。

Cannon算法的基本思想[3-4]是：把矩阵***A***、矩阵***B***划分 $p\times p$ 个方块，分别对应算法中的 $p\times p$ 个进程，最初每个进程分别有矩阵***A***、矩阵***B***的一个子矩阵块，即把子矩阵$\boldsymbol{A}_{i,j}$、矩阵$\boldsymbol{B}_{i,j}$分配给进程 $P_{i,j}(0\le i,j<p)$。每个进程对子矩阵进行乘法运算，得到结果矩阵$\boldsymbol{C}_{i,j}$，然后矩阵***A***的每个子矩阵块向左移动一步(把一行看作首尾相连成一个环)，矩阵***B***的每个子矩阵块向上移动一步(同样把一列看作首尾相连的一个环)，进程再对新得到子矩阵块进行乘法运算把结果累加到矩阵$\boldsymbol{C}_{i,j}$。这样总共进行 p 步后，把结果矩阵块汇总即为计算结果。

Cannon算法在并行系统中的处理机数量恰好为 $p\times p$ 时，能得到理想的效果。一般情况下，系统中的处理机数量不会恰好等于 $p\times p$，而应小于 $p\times p$，文献[5]对矩阵相乘的Cannon算法进行了改进，将属于同一列的2个相邻子任务优先考虑分配在同一个工作站上运行。并在PVM并行计算编程环境下实现了该算法(算法1)。

算法1 处理机数量k和子任务(进程)数P^2的关系为 $(p-1)^2<k<p^2$。

机群系统中有 (p^2-k) 台工作站需运行两个子任务，其余工作站上只运行一个子任务[5]。在Cannon算法中，每个进程要完成的任务是一样的，先计算本进程中的两个子矩阵块的乘法，再与相邻的进程交换数据，每个进程的工作量是一样的。由于进程间要交换数据，因此要求各进程同步运行。这样，只分配到一个进程的工作站运行完后，就必须等待运行2个进程的工作站，在其两个子任务都运行结束后，才能进行到下一步的通信和计算。在机群中的各工作站性能相差不大时，则每一步运算所花费的时间是理论时间的两倍。

进一步分析算法1，同时运行两个子任务的工作站数量占所有工作站的比率为$(p^2-k)/k$，因为 $(p-1)^2<k<p^2$，所以在 $k=(p-1)^2+1$，比率达到最大值，即

$$\frac{p^2-k}{k}\le\frac{p^2-((p-1)^2+1)}{(p-1)^2+1}=\frac{2p-2}{p^2-2p+2}$$

当 $p\ge5$ 时，这个值小于0.5，且k越接近P^2时，值越小。

作者简介：韦安定(1977－)，男，硕士研究生，主研方向：网络并行计算；李代平，教授；文 剑，硕士研究生

收稿日期：2006-12-27 **E-mail：**andy21st@163.com

因为在工作站数量 $p^2 \geq 4$，所以cannon算法才会具有较好的优越性。表 1 给出了P为 3, 4 时，k取不同值的比率。

表 1　k 取不同值的比率

p	k	$\frac{p^2-k}{k}$
3	5	0.80
3	6	0.50
3	7~8	<0.29
4	10	0.60
4	11	0.45
4	12～15	<0.34

在上述算法中，由于运行两个任务的工作站一般情况下只占总工作站数量中很小的一部分，因此机群系统中，多数处理机在大部分时间里都是处于空闲状态。系统中的资源并没能很好地利用。

2　矩阵 *A* 按行块划分的并行算法

上述可见，在工作站数量小于P^2时，即工作站数量不是某个数的平方时，Cannon算法在PVM环境下并不能得到比较好的性能。这也是本文讨论问题的前提条件，应为在工作站数量为P^2时，Cannon无疑是个理想的算法。PVM环境是消息传递机制的并行编程环境，各子任务间的数据交换通过消息传递的方式实现，由于PVM中的消息传递开销较大，因此应尽量减少子任务间的消息交换。且PVM环境下的并行算法比较适合较大的任务粒度，这样才能得到较高的效率。本文的矩阵乘法的并行算法(算法 2)正是基于这样的考虑。

算法 2 观察矩阵乘法中，矩阵 ***A*** 的一行与矩阵 ***B*** 相乘得到结果矩阵 ***C*** 的一行，矩阵 ***A*** 的 r 行与矩阵 B 相乘得到 ***C*** 的 r 行。可以把矩阵 ***A*** 按行划分成若干行块，每个子任务运算一个行块与矩阵 ***B*** 的乘法，得到结果矩阵 ***C*** 的一部分。这样，除了主任务与各个子任务之间有消息传递外，各个子任务间没有消息传递，这样就可以减少消息的传递次数，较少通信开销。PVM 支持消息的广播机制，可以把矩阵 ***B*** 以广播消息的方式发送给各个子任务，对于矩阵 ***A***，则根据机群系统中的工作站数量 k 按行平均划分成 k 子矩阵块，每个子矩阵块含有矩阵 ***A*** 的 n/k 行(对于 n 不能被 k 整除的情况，例如其余数为 d，可做如下处理：前 d 个子矩阵为 $n/k+1$行，为方便起见，这里假设 n 能被 k 整除)。平均第 i 个子任务含有矩阵 ***A*** 的第 $i\cdot\frac{n}{k}$ 行~第 $(i+1)\cdot\frac{n}{k}$ 行($i=0,1,\cdots)$，然后与矩阵 ***B*** 相乘的到结果矩阵 ***C*** 的第 $i\cdot\frac{n}{k}$ (i-1)行~第 $i\cdot\frac{n}{k}$ 行。这样每个任务除了从主进程接收数据并向主进程返回结果外，与其他子任务间没有消息的交换，这样也可以减少消息的传递的次数。

以下分析算法 2 的性能。为不失一般性，以 2 个n阶方阵相乘为例进行其性能分析。该算法的性能分析包括传递消息的时间t_{comm}和各子任务中的计算时间t_{comp}。

算法中消息的传递有：整个矩阵***B***的广播，传递矩阵***A***的各个分快，子进程向主进程返回结果。令t_s为传递消息的启动时间；t_w为传递一个单位消息的时间，这里表示传递一个矩阵元素的时间。在PVM环境下，任务将要广播的消息发送给PVMD，路由层依次将每个包拷贝给本地的任务以及远程的目标PVMD，而目标则依次将每个包拷贝给每个目标任务。可见，广播消息可以减少消息传递多次启动的时间，忽略本地任务的通信，令t_b为广播一个单位消息的时间，则广播矩阵***B***的时间为 $t_s+t_bn^2$。

发送矩阵 ***A*** 的一个子矩阵块的时间为

$$t_s+t_wn^2/k$$

除了在本地运行的子任务，另外的要给远程任务发送 k-1 个子矩阵块，返回结果的子矩阵块的消息传递时间和传递各个子矩阵块给子任务的时间开销为

$$2(k-1)(t_s+t_wn^2/k)$$

总的消息传递时间为

$$t_{\text{comm}}=(2k-1)t_s+\frac{2(k-1)}{k}n^2t_w+n^2t_b$$

计算时间，每个任务的计算量是一样的，$(n/k)\times n$ 阶矩阵与 n 阶方阵相乘，按一般的串行算法，假设一次加法和一次乘法的计算时间为 T，矩阵块的运算时间为

$$t_{\text{comp}}=T\cdot n^3/k$$

下面对算法 1 进行性能分析。按前文所述，划分成P^2个子任务，即矩阵***A***、矩阵***B***矩阵分别被划分成 $(n/p)\times(n/p)$ 的矩阵。程序开始时，要先初始化，即要把各子矩阵块分配至对应的子任务，每个任务各分配到矩阵***A***、矩阵***B***的一个子矩阵块，初始化过程的通信时间为

$$2(p^2-1)(t_s+t_wn^2/p^2)$$

在每一步计算后，子任务要与相邻的子任务交换矩阵 ***A***、矩阵 ***B*** 的子矩阵块，每步的通信时间为

$$2(t_s+t_wn^2/p^2)$$

这个过程还要经过 $(p-1)$ 步，交换消息的时间为

$$2(p-1)(t_s+t_wn^2/p^2)$$

还要将各个子任务的计算结果汇总，在汇总的过程中，各个子任务依次与负责收集结果的任务通信，通信时间为

$$(p^2-1)(t_s+t_wn^2/p^2)$$

总的通信时间为

$$t_{\text{comm}}=(3p^2+2p-5)t_s+\frac{3p^2+2p-5}{p^2}n^2t_w$$

每个子进程要进行 P 次 $(n/p)\times(n/p)$ 阶的子矩阵的乘法运算，总的运算时间为

$$t_{\text{comp}}=n^3/p^2\cdot T$$

这是在机群系统中的工作站数量与进程数P^2相等时的理论计算时间，在 $(p-1)^2<k<p^2$ 时，因为至少有一台工作站运行两个子任务，所以计算时间为

$$t_{\text{comp}}=2n^3/p^2\cdot T$$

对比两种算法的性能分析，在算法 2 中，要广播整个矩阵 ***B***，通信开销时间要比算法 1 大些，但在计算时间方面，由于各子任务间要同步，因此算法 1 中的每一步的计算时间至少是算法 2 的 2 倍。虽然算法 2 的通信时间较大，但通信时间要小于计算时间，运行效率要高于算法 1。

3　算法在 PVM 并行编程环境下的实现

算法采用 master/slave 模式来实现，将矩阵 ***B*** 整体用多播模式传送给每个 Slave 进程，根据实际情况可划分成 *ntask* 个进程，把矩阵 ***A*** 的各个划分数据分配给 Slave 进程。Slave 进程接收消息对应于 master 发送的消息，反之，返回结果对应于 master 进程的接收结果段，master 进程的主要伪代码如下：

```
//Master 进程
pvm_spawn( );//派生任务
用 pvm_mcast()函数广播矩阵 B 给各子任务;
 for (i=0;i<ntask;i++)
 {//分配数据到各个任务
  发送矩阵 A 的起始行标记 offset;
发送要分配的行数 rows;
```

(下转第 77 页)