

## 1.1 LU 分解

从本小节起我们将对 LU 分解等矩阵分解的并行计算做一些简单讨论。在许多应用问题的科学计算中,矩阵的 LU 分解是基本、常用的一种矩阵运算,它是求解线性方程组的基础,尤其在解多个同系数阵的线性方程组时特别有用。

### 1.1.1 矩阵的 LU 分解及其串行算法

对于一个  $n$  阶非奇异方阵  $A=[a_{ij}]$ , 对  $A$  进行 LU 分解是求一个主对角元素全为 1 的下三角方阵  $L=[l_{ij}]$  与上三角方阵  $U=[u_{ij}]$ , 使  $A=LU$ 。设  $A$  的各阶主子行列式皆非零,  $U$  和  $L$  的元素可由下面的递推式求出:

$$\begin{aligned} a_{ij}^{(1)} &= a_{ij} \\ a_{ij}^{(k+1)} &= a_{ij}^{(k)} - l_{ik} u_{kj} \\ l_{ik} &= \begin{cases} 0 & i < k \\ 1 & i = k \\ a_{ik}^{(k)} u_{kk}^{-1} & i > k \end{cases} \\ u_{kj} &= \begin{cases} 0 & k > j \\ a_{kj}^{(k)} & k \leq j \end{cases} \end{aligned}$$

在计算过程中,首先计算出  $U$  的第一行元素,然后算出  $L$  的第一列元素,修改相应  $A$  的元素;再算出  $U$  的第二行,  $L$  的第二列 $\cdots$ ,直至算出  $u_{nn}$  为止。若一次乘法和加法运算或一次除法运算时间为一个单位时间,则下述 LU 分解的串行算法 18.9 时间复杂度为

$$\sum_{i=1}^n (i-1)i = (n^3 - n)/3 = O(n^3)。$$

#### 算法 18.9 单处理器上矩阵 LU 分解串行算法

输入: 矩阵  $A_{n \times n}$

输出: 下三角矩阵  $L_{n \times n}$ , 上三角矩阵  $U_{n \times n}$

Begin

```
(1)for k=1 to n do
  (1.1)for i=k+1 to n do
     $a[i,k]=a[i,k]/a[k,k]$ 
  end for
  (1.2)for i=k+1 to n do
    for j=k+1 to n do
       $a[i,j]=a[i,j]-a[i,k]*a[k,j]$ 
    end for
  end for
end for
(2)for i=1 to n do
  (2.1)for j=1 to n do
    if ( $j < i$ ) then
       $l[i,j]=a[i,j]$ 
    end if
  end for
end for
```

```

else
    u[i,j]=a[i,j]
end if
end for
(2.2)l[i,i]=1
end for
End

```

### 1.1.2 矩阵 LU 分解的并行算法

在 LU 分解的过程中，主要的计算是利用主行  $i$  对其余各行  $j$ , ( $j > i$ ) 作初等行变换，各行计算之间没有数据相关关系，因此可以对矩阵  $A$  按行划分来实现并行计算。考虑到在计算过程中处理器之间的负载均衡，对  $A$  采用行交叉划分：设处理器个数为  $p$ ，矩阵  $A$  的阶数为  $n$ ， $m = \lceil n/p \rceil$ ，对矩阵  $A$  行交叉划分后，编号为  $i$  ( $i=0,1,\dots,p-1$ ) 的处理器存有  $A$  的第  $i, i+p, \dots, i+(m-1)p$  行。然后依次以第  $0,1,\dots,n-1$  行作为主行，将其广播给所有处理器，各处理器利用主行对其部分行向量做行变换，这实际上是各处理器轮流选出主行并广播。若以编号为  $\text{my\_rank}$  的处理器的主行元素作为主行，并将它广播给所有处理器，则编号大于等于  $\text{my\_rank}$  的处理器利用主行元素对其第  $i+1, \dots, m-1$  行数据做行变换，其它处理器利用主行元素对其第  $i, \dots, m-1$  行数据做行变换。具体并行算法框架描述如下：

#### 算法 18.10 矩阵 LU 分解的并行算法

输入：矩阵  $A_{n \times n}$

输出：下三角矩阵  $L_{n \times n}$ , 上三角矩阵  $U_{n \times n}$

Begin

对所有处理器  $\text{my\_rank}$  ( $\text{my\_rank}=0, \dots, p-1$ ) 同时执行如下的算法：

for  $i=0$  to  $m-1$  do

for  $j=0$  to  $p-1$  do

(1) if ( $\text{my\_rank}=j$ ) then /\*当前处理的主行在本处理器\*/

(1.1)  $v=i*p+j$  /\*  $v$  为当前处理的主行号\*/

(1.2) for  $k=v$  to  $n$  do

$f[k]=a[i,k]$  /\* 主行元素存到数组  $f$  中\*/

end for

(1.3) 向其它所有处理器广播主行元素

else /\*当前处理的主行不在本处理器\*/

(1.4)  $v=i*p+j$

(1.5) 接收主行所在处理器广播来的主行元素

end if

(2) if ( $\text{my\_rank} \leq j$ ) then

(2.1) for  $k=i+1$  to  $m-1$  do

(i)  $a[k,v]=a[k,v]/f[v]$

(ii) for  $w=v+1$  to  $n-1$  do

$a[k,w]=a[k,w]-f[w]*a[k,v]$

end for

end for

else

(2.2) for  $k=i$  to  $m-1$  do

```

(i)  $a[k,v]=a[k,v]/f[v]$ ;
(ii) for  $w=v+1$  to  $n-1$  do
     $a[k,w]=a[k,w]-f[w]*a[k,v]$ 
end for
end for
end if
end for
end for
End

```

计算完成后，编号为 0 的处理器收集各处理器中的计算结果，并从经过初等行变换的矩阵  $A$  中分离出下三角矩阵  $L$  和上三角矩阵  $U$ 。若一次乘法和加法运算或一次除法运算时间为一个单位时间，则计算时间为  $(n^3-n)/3p$ ；又  $n-1$  行数据依次作为主行被广播，通信时间为  $(n-1)(t_s+nt_w)\log p = O(n \log p)$ ，因此并行计算时间  $T_p = (n^3-n)/3p + (n-1)(t_s+nt_w)\log p$ 。

MPI 源程序请参见章末附录。

## 1.2 QR 分解

$A=[a_{ij}]$  为一个  $n$  阶实矩阵，对  $A$  进行 QR 分解，就是求一个非奇异(Nonsingular)方阵  $Q$  与上三角方阵  $R$ ，使得  $A=QR$ 。其中方阵  $Q$  满足： $Q^T=Q^{-1}$ ，称为正交矩阵(Orthogonal Matrix)，因此 QR 分解又称为正交三角分解。

### 1.2.1 矩阵 QR 分解的串行算法

对  $A$  进行正交三角分解，可用一系列平面旋转矩阵左乘  $A$ ，以使  $A$  的下三角元素逐个地被消为 0。设要消去的下三角元素为  $a_{ij}(i>j)$ ，则所用的平面旋转方阵为：

$$Q_{ij} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & c & s & \\ & & -s & c & \\ & & & & 1 \\ & & & & & 1 \end{bmatrix}$$

其中，除了  $(i,i)(j,j)$  元素等于  $c$ ， $(i,j)$  元素与  $(j,i)$  元素分别等于  $-s$  与  $s$  以外，其余元素皆与单位方阵  $I$  的对应元素相等，而  $c,s$  由下式计算：

$$c = \cos \theta = a_{jj} / \sqrt{a_{jj}^2 + a_{ij}^2}$$

$$s = \sin \theta = a_{ij} / \sqrt{a_{jj}^2 + a_{ij}^2}$$

其中， $\theta$  为旋转角度。这样，在旋转后所得到的方阵  $A'$  中，元素  $a_{ij}'$  为 0， $A'$  与  $A$  相比仅在第  $i$  行、第  $j$  行上的元素不同：

$$a_{jk}' = c \times a_{jk} + s \times a_{ik}$$

$$a'_{ik} = -s \times a_{jk} + c \times a_{ik} \quad (k = 1, 2, \dots, n)$$

消去  $A$  的  $r=n(n-1)/2$  个下三角元素后得到上三角阵  $R$ ，实际上是用  $r$  个旋转方阵  $Q_1, Q_2, \dots, Q_r$  左乘  $A$ ，即：

$$R = Q_r Q_{r-1} \cdots Q_1 A$$

设  $Q_0 = Q_r Q_{r-1} \cdots Q_1$ ，易知  $Q_0$  为一正交方阵，则有：

$$R = Q_0 A$$

即

$$A = Q_0^{-1} R = Q_0^T R = QR$$

其中  $Q = Q_0^{-1} = Q_0^T$  为一正交方阵，而  $Q_0$  可以通过对单位阵  $I$  进行同样的变换而得到，这样可得到  $A$  的正交三角分解。QR 分解串行算法如下：

#### 算法 18.11 单处理器上矩阵的 QR 分解串行算法

输入：矩阵  $A_{n \times n}$ ，单位矩阵  $Q$

输出：矩阵  $Q_{n \times n}$ ，矩阵  $R_{n \times n}$

Begin

(1) for  $j=1$  to  $n$  do

for  $i=j+1$  to  $n$  do

(i)  $sq = \sqrt{a[j,j]^2 + a[i,j]^2}$

$c = a[j,j]/sq$

$s = a[i,j]/sq$

(ii) for  $k=1$  to  $n$  do

$aj[k] = c * a[j,k] + s * a[i,k]$

$qj[k] = c * q[j,k] + s * q[i,k]$

$ai[k] = -s * a[j,k] + c * a[i,k]$

$qi[k] = -s * q[j,k] + c * q[i,k]$

end for

(iii) for  $k=1$  to  $n$  do

$a[j,k] = aj[k]$

$q[j,k] = qj[k]$

$a[i,k] = ai[k]$

$q[i,k] = qi[k]$

end for

end for

end for

(2)  $R = A$

(3) MATRIX\_TRANSPOSITION( $Q$ ) /\* 对  $Q$  实施算法 18.1 的矩阵转置 \*/

End

算法 18.11 要对  $n(n-1)/2$  个下三角元素进行消去，每消去一个元素要对两行元素进行旋转变换，而对一个元素进行旋转变换又需要 4 次乘法和 2 次加法，所以若取一次乘法或一次加法运算时间为一个单位时间，则该算法的计算时间为  $n(n-1)/2 * 2n * 6 = 6n^2(n-1) = O(n^3)$ 。

### 1.2.2 矩阵 QR 分解的并行算法

由于 QR 分解中消去  $a_{ij}$  时，同时要改变第  $i$  行及第  $j$  行两行的元素，而在 LU 分解中，仅利用主行  $i(i < j)$  变更第  $j$  行的元素。因此 QR 分解并行计算中对数据的划分与分布与 LU 分解就不一样。设处理器个数为  $p$ ，对矩阵  $A$  按行划分为  $p$  块，每块含有连续的  $m$  行向量，

$m = \lceil n/p \rceil$ , 这些行块依次记为  $A_0, A_1, \dots, A_{p-1}$ , 分别存放在标号为  $0, 1, \dots, p-1$  的处理器中。

在 0 号处理器中, 计算开始时, 以第 0 行数据作为主行, 依次和第  $0, 1, \dots, m-1$  行数据做旋转变换, 计算完毕将第 0 行数据发送给 1 号处理器, 以使 1 号处理器将收到的第 0 行数据作为主行和自己的  $m$  行数据依次做旋转变换; 在此同时, 0 号处理器进一步以第 1 行数据作为主行, 依次和第  $2, 3, \dots, m-1$  行数据做旋转变换, 计算完成后将第 1 行数据发送给 1 号处理器; 如此循环下去。一直到以第  $m-1$  行数据作为主行参与旋转变换为止。

在 1 号处理器中, 首先以收到的 0 号处理器的第 0 行数据作为主行和自己的  $m$  行数据做旋转变换, 计算完将该主行数据发送给 2 号处理器; 然后以收到的 0 号处理器的第 1 行数据作为主行和自己的  $m$  行数据做旋转变换, 再将该主行数据发送给 2 号处理器; 如此循环下去, 一直到以收到的 0 号处理器的第  $m-1$  行数据作为主行和自己的  $m$  行数据做旋转变换并将第  $m-1$  行数据发送给 2 号处理器为止。然后, 1 号处理器以自己的第 0 行数据作为主行, 依次和第  $1, 2, \dots, m-1$  行数据做旋转变换, 计算完毕将第 0 行数据发送给 2 号处理器; 接着以第 1 行数据作为主行, 依次和第  $2, 3, \dots, m-1$  行数据做旋转变换, 计算完毕将第 1 行数据发送给 2 号处理器; 如此循环下去。一直到以第  $m-1$  行数据作为主行参与旋转变换为止。除了  $p-1$  号处理器以外的所有处理器都按此规律先顺序接收前一个处理器发来的数据, 并作为主行和自己的  $m$  行数据作旋转变换, 计算完毕将该主行数据发送给后一个处理器。然后依次以自己的  $m$  行数据作主行对主行后面的数据做旋转变换, 计算完毕将主行数据发给后一个处理器。

在  $p-1$  号处理器中, 先以接收到的数据作为主行参与旋转变换, 然后依次以自己的  $m$  行数据作为主行参与旋转变换。所不同的是,  $p-1$  号处理器作为最后一个处理器, 负责对经过计算的全部数据做汇总。具体并行算法框架描述如下:

#### 算法 18.12 矩阵 QR 分解并行算法

**输入:** 矩阵  $A_{n \times n}$ , 单位矩阵  $Q$

**输出:** 矩阵  $Q_{n \times n}$ , 矩阵  $R_{n \times n}$

**Begin**

对所有处理器 my\_rank(my\_rank=0, ..., p-1) 同时执行如下的算法:

(1) if (my\_rank=0) then /\*0 号处理器\*/

(1.1) for  $j=0$  to  $m-2$  do

(i) for  $i=j+1$  to  $m-1$  do

Turnningtransform() /\*旋转变换\*/

end for

(ii) 将旋转变换后的 A 和 Q 的第  $j$  行传送到第 1 号处理器

end for

(1.2) 将旋转变换后的 A 和 Q 的第  $m-1$  行传送到第 1 号处理器

end if

(2) if ((my\_rank>0) and (my\_rank<(p-1))) then /\*中间处理器\*/

(2.1) for  $j=0$  to my\_rank\*m-1 do

(i) 接收左邻处理器传送来的 A 和 Q 子块中的第  $j$  行

(ii) for  $i=0$  to  $m-1$  do

Turnningtransform() /\*旋转变换\*/

end for

(iii) 将旋转变换后的 A 和 Q 子块中的第  $j$  行传送到右邻处理器

end for

```

(2.2) for j=0 to m-2 do
    (i)z=my_rank*m
    (ii)for i=j+1 to m-1 do
        Turnningtransform() /*旋转变换*/
    end for
    (iii)将旋转变换后的 A 和 Q 子块中的第 j 行传送到右邻处理器
end for
(2.3)将旋转变换后的 A 和 Q 子块中的第 m-1 行传送到右邻处理器
end if
(3)if (my_rank=(p-1)) then /*p-1 号处理器*/
    (3.1) for j=0 to my_rank*m-1 do
        (i)接收左邻处理器传送来的 A 和 Q 子块中的第 j 行
        (ii)for i=0 to m-1 do
            Turnningtransform() /*旋转变换*/
        end for
    end for
    (3.2)for j=0 to m-2 do
        for i=j+1 to m-1 do f
            Turnningtransform() /*旋转变换*/
        end for
    end for
end if
End

```

当所有的计算完成后，编号为  $p-1$  的处理器负责收集各处理器中的计算结果， $R$  矩阵为经旋转变换的  $A$  矩阵， $Q$  矩阵为经旋转变换的  $Q$  的转置。QR 分解并行计算的时间复杂度分析因每个处理器的开始计算时间不同而不同于其它算法，每个处理器都要顺序对其局部存储器中的  $m$  行向量做自身的旋转变换，其时间用  $T_{computer}$  表示。以 16 阶矩阵为例，4 个处理器对其进行 QR 分解的时间分布图如图 18.4 所示，这里  $i^{(j)}$  表示以接收到的第  $j$  号处理器的第  $i$  行数据作为主行和自己的  $m$  行数据做旋转变换。 $\Delta t$  表示第  $p-1$  号处理器与第 0 号处理器开始计算时间的间隔。

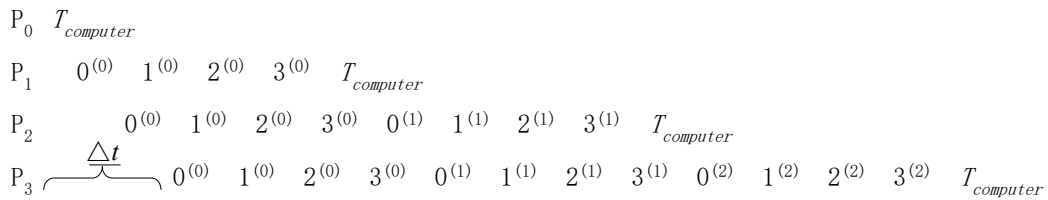


图 18.1 QR 分解并行计算的时间分布图

此处不妨以第  $p-1$  号处理器为对象分析并行计算时间，若取一次乘法或一次加法运算时间为一个单位时间，由于对一个元素进行旋转变换需要 4 次乘法和 2 次加法，因而需要六个单位时间。要对  $m(m-1)/2$  个下三角元素进行消去，每消去一个元素要对两行元素进行旋转变换，故自身的旋转变换时间  $T_{computer}=6mn(m-1)$ ；第  $p-1$  号处理器依次接收前  $n-m$  行数据作为主行参与旋转变换，其计算时间为  $(n-m)*m*2n*6=12nm(n-m)$ ，通信时间为  $2(n-m)(t_s+nt_w)$ ；时间间隔  $\Delta t$  即第  $p-1$  号处理器等待第 0 号处理器第 0 行数据到达的时间，第 0 行数据在第 0

号处理器中参与 $(m-1)$ 行旋转变换, 然后被发送给第 1 号处理器, 这段计算时间为 $(m-1)*2n*6$ , 通信时间为  $2(t_s+nt_w)$ ; 接着第 0 行数据经过中间 $(p-2)$ 个处理器的旋转变换被发送给第  $p-1$  号处理器, 这段计算时间为 $(p-2)m*2n*6$ , 通信时间为  $4(p-2)(t_s+nt_w)$ , 所以 $\Delta t=12n(mp-m-1)+(4p-6)(t_s+nt_w)$ , 并行计算时间为  $T_p=12n^2-18mn-6nm^2+12n^2m-12n+(2n-2m+4p-6)(t_s+nt_w)$ 。