第十讲：矩阵计算并行算法

陈俊清

jqchen@math.tsinghua.edu.cn

清华大学数学科学系

May 9, 2013

Part I

稀疏矩阵并行算法

Outline

1. 稀疏矩阵向量乘并行计算

2. 稀疏矩阵乘并行计算

Outline

1. 稀疏矩阵向量乘并行计算
2. 稀疏矩阵乘并行计算

稀疏矩阵

稀疏矩阵的存储格式：

- Aij格式:需要三个相同长度的数组，一个存放矩阵元素，一个存放对应元素的行指标，一个存放对应元素的列指标， 矩阵元素可以不要求按顺序存放。
- CSR/CSC格式（压缩稀疏行/列格式）：行压缩格式同样需三个数组，一个存放非零元， 一个整数数组标识非零元所在的列指标，一个较小的数组存放每一行中第一个非零元在前两个数组中的位置。

稀疏矩阵（续）

例：

$$A = \begin{pmatrix} 1 & & & & & & & \\ & 2 & & & 3 & & & \\ 4 & 5 & 6 & 7 & & & & 8 \\ & & & 9 & & & & \\ & & & & 10 & & & \\ & & 11 & & & 12 & & \\ & & & & & 13 & 14 & \\ & & 15 & 16 & & & & 17 \end{pmatrix}$$

Aij格式：

$$x = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17];$$
$$x^{(I)} = [1\ 2\ 2\ 3\ 3\ 3\ 3\ 3\ 4\ 5\ 6\ 6\ 7\ 7\ 8\ 8\ 8];$$
$$x^{(J)} = [1\ 2\ 5\ 1\ 2\ 3\ 4\ 8\ 4\ 5\ 3\ 6\ 6\ 7\ 3\ 4\ 8];$$

稀疏矩阵（续）

CSR格式：

$$x = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17]$$
$$x^{(J)} = [1\ 2\ 5\ 1\ 2\ 3\ 4\ 8\ 4\ 5\ 3\ 6\ 6\ 7\ 3\ 4\ 8]$$
$$x^{(R)} = [1\ 2\ 4\ 9\ 10\ 11\ 13\ 15\ 18]$$

A的第i行的非零元为：

$$x_j, j = x_i^{(R)}, x_i^{(R)} + 1, ..., x_{i+1}^{(R)} - 1$$

这些非零元的列指标为：

$$x_j^{(J)}, j = x_i^{(R)}, x_i^{(R)} + 1, ..., x_{i+1}^{(R)} - 1$$

注：x^R的最后一个元素可以不要，根据前两个数组长度就可以知道非零元的个数。

稀疏矩阵向量乘的串行算法

$n \times n$矩阵A采用CSR格式存储到$\mathbf{x}, \mathbf{x}^{(J)}, \mathbf{x}^{(R)}$中， A与稠密向量$\mathbf{z}$相乘的第i个分量的计算公式为：

$$y_i = \sum_{k=x_i^{(R)}}^{x_{i+1}^{(R)}-1} x_k z_{x_k^{(J)}}$$

算法9.13 一般CSR格式下稀疏矩阵向量乘的串行算法

1. Set $\mathbf{y} = \mathbf{0}$;
2. for i=1 to n do
   for $k = x_i^{(R)}$ to $k = x_{i+1}^{(R)} - 1$ do
   $y_i = y_i + x_k z_{x_k^{(J)}}$;
   endfor
   endfor

如果A对称，且只采用CSR格式存储了矩阵的上三角部分中的非零元，则在计算完成算法9.13后，还需要一步计算：

$$y_{x_k^{(J)}} \leftarrow y_{x_k^{(J)}} + x_k z_i, k = x_i^{(R)} + 1, ..., x_{i+1}^{(R)} - 1, i = 1, 2, ..., n.$$

算法9.14 对称稀疏矩阵向量乘的串行算法

1. Set $\mathbf{y} = \mathbf{0}$;
2. for i=1 to n do
   for $k = x_i^{(R)}$ to $k = x_{i+1}^{(R)} - 1$ do $y_i = y_i + x_k z_{x_k^{(J)}}$;
   endfor
3. for i=1 to n do
   for $k = x_i^{(R)} + 1$ to $x_{i+1}^{(R)} - 1$ do $y_{x_k^{(J)}} = y_{x_k^{(J)}} + x_k z_i$;
   endfor

CSR格式下稀疏矩阵向量乘的并行算法

矩阵A为$n \times n$阶CSR格式存储的稀疏矩阵，其非零元个数为：

$$l = x^{(R)}_{n+1} - 1.$$

考虑利用行划分进行并行算法设计，由于每行非零元素个数不同，而每行计算量与非零元素个数成正比， 为保证负载平衡，在进行行划分时应尽量让每块包含非零元尽量相等。

最理想状况是l个非零元平均分配到各个进程上，但这要在CSR格式所采用的数组之外引入额外的辅助数组才能实现， 所以我们依然采用行划分方式，在此前提下，使得每块的非零元尽可能相等。

设p为进程数，我们将行分配算法描述如下：

算法9.15 矩阵行的分配算法

1. Set $j = 0; q = 0; l = x_{n+1}^{(R)} - 1; s_0 = 1;$
2. for i=1 to n do

    Compute $j = j + x_{i+1}^{(R)} - x_i^{(R)};$

    if $j > (q+1)l/p$ then

        Set $s_{q+1} = i; q = q + 1;$

    endif

    Set $s_p = n + 1;$

   endfor

算法9.15结束后，对$i = 0, 1, ..., p - 1$,可以将矩阵的s_i到$s_{i+1} - 1$行分配到进程P_i,这些行组成的块看成一个单独的矩阵， 在进程P_i上采用CSR格式存储在$\mathbf{x}, \mathbf{x}^{(J)}, \mathbf{x}^{(R)}$中， 向量$\mathbf{y}, \mathbf{z}$分块为：

$$\mathbf{y} = \begin{pmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \cdots \\ \mathbf{y}_{p-1} \end{pmatrix} \mathbf{z} = \begin{pmatrix} \mathbf{z}_0 \\ \mathbf{z}_1 \\ \cdots \\ \mathbf{y}_{p-1} \end{pmatrix}$$

$\mathbf{y}_i, \mathbf{z}_i$含有的分量个数均为$s_{i+1} - s_i$,分别存在进程$P_i$的$\mathbf{y}, \mathbf{z}$中。

CSR格式下稀疏矩阵向量乘的并行算法（续）

通信结构：

- 在每个进程上，用$\mathbf{w}^{(J)}$记录依赖于其它进程的$\mathbf{z}$中分量的标号，用$w_j^{(R)}$记录这些依赖分量中存储在进程j上的首分量在$\mathbf{w}^{(J)}$中的位置；
- $\mathbf{w}^{(J)}$的长度不超过n，$\mathbf{w}^{(R)}$的长度为p+1；
- 每个进程依赖于进程j的分量个数为$l_j = w_{j+1}^{(R)} - w_j^{(R)}$；
- 多对多广播l_j，确定每个进程给其它进程发送分量个数；
- 将每个进程上$\mathbf{w}^{(J)}$中的第j（$j = 0, 1, ..., p-1$）段指标传给进程j，就可以得到每个进程需向其它进程发送哪些分量了。

算法9.16 CSR存储格式下稀疏矩阵向量乘的并行算法
略。
注：可以多对多广播向量$\mathbf{z}$来减少上述分析中的复杂判断，但通信量会相应增加。

Outline

如果将矩阵操作分解为向量操作，则实际上要涉及的向量操作包括向量相加与向量内积两种。对稀疏矩阵乘法，所涉及的基本向量操作也是这两种。

对一个n维向量，如果其中只有少量几个分量不为0，就称之为一个稀疏向量。设**x**非零元对应的指标集合为$S = \{i_k : k = 1, 2, ..., m\}$，则**x**可被存储为：

$$\mathbf{y} = [x_{i_1}, x_{i_2}, ..., x_{i_m}]^T, \mathbf{z} = [i_1, i_2, ..., i_m]^T;$$

设有两个n维稀疏向量$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}$,分别存储为：

$$\mathbf{y}^{(1)} = [x_{i_1}^{(1)}, x_{i_2}^{(1)}, ..., x_{i_m}^{(1)}]^T, \mathbf{z}^{(1)} = [i_1, i_2, ..., i_m]^T$$
$$\mathbf{y}^{(2)} = [x_{j_1}^{(2)}, x_{j_2}^{(2)}, ..., x_{j_l}^{(2)}]^T, \mathbf{z}^{(2)} = [j_1, j_2, ..., j_l]^T$$

稀疏向量的加法

算法9.17 稀疏向量相加的串行算法

1. for j=1 to l do $w_{z_j^{(2)}} = y_j^{(2)}$;
2. for i=1 to m do

   $y_i = y_i^{(1)}, z_i = z_i^{(1)}$;

   if $w_{z_i} \neq 0$ then

   $y_i = y_i + w_{z_i}$;Set $w_{z_i} = 0$;

   endif

   endfor
3. for j=1 to l do

   if $w_{z_j^{(2)}} \neq 0$ then

   $i = i + 1; y_i = y_j^{(2)}; z_i = z_j^{(2)}$;

   endif

   endfor

稀疏向量的内积

对前述稀疏向量$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}$,若要计算其内积，可以先将$\mathbf{y}^{(2)}$扩展为n维向量$\mathbf{w}$，即令$w_{z_k^{(2)}} = y_k^{(2)}$，再计算:

$$(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \sum_{k=1}^{m} y_k^{(1)} w_{z_k^{(1)}}$$

具体算法为:

算法9.18 稀疏矩阵内积

1. for j=1 to l do $w_{z_j^{(2)}} = y_j^{(2)}$;
2. Set d=0;
3. for i=1 to m do $d = d + y_i^{(1)} w_{z_i^{(1)}}$.

稀疏矩阵的串行乘法

A、B为两个n阶矩阵，分别按CSR格式存为$\mathbf{x}, \mathbf{x}^{(J)}, \mathbf{x}^{(R)}$和$\mathbf{y}, \mathbf{y}^{(J)}, \mathbf{y}^{(R)}$,计算结果存在$\mathbf{z}, \mathbf{z}^{(J)}, \mathbf{z}^{(R)}$中。

$$c_{i,0:n-1} = \sum_{k=0}^{n-1} a_{i,k} b_{k,0:n-1}$$

所以每一个$a_{i,k}$都进行第i行的一个部分和的计算。
由于A,B均按CSR格式，乘积也按该格式，故其乘积计算要按其结果一行一行地进行，以在必要的时候增加非零元。 基于此，可得到如下的算法:

算法9.19, CSR存储格式下稀疏矩阵乘的串行算法

1. Set $n_z = 1; z_0^{(R)} = 1;$
2. for i=0 to n-1 do $w_i = 0;$
3. for i=0 to n-1 do

    Set l=0;

    for $i_k = x_i^{(R)}$ to $x_{i+1}^{(R)} - 1$ do (矩阵A的第i行)

        Set $k = x_{i_k}^{(J)};$ （A的i行k列）

        for $k_j = y_k^{(R)}$ to $y_{k+1}^{(R)} - 1$ do （B的第k行）

            Set $j = x_{k_j}^{(J)};$ （B的第j列）

            if $w_j \neq 0$ then $w_j = w_j + x_{i_k} y_{k_j};$

            else $w_j = x_{i_k} y_{k_j}; l = l + 1; w_l^{(J)} = j;$ （增加非零元）

            endif

        endfor

    endfor

CSR格式串行算法

for k=1 to l do

$j = w_k^{(J)}; z_{n_z}^{(J)} = j; z_{n_z} = w_j;$

$n_z = n_z + 1; w_j = 0;$

endfor

$z_{i+1}^{(R)} = n_z;$

endfor

行行划分下的稀疏矩阵的并行乘法

考虑前述矩阵A，B在p个进程上并行算法设计。

只考虑行行划分下的算法，并且忽略负载平衡的影响，假设n能被p整除，且记$m = n/p$,即每个进程上包含m行。

每个进程上对应的A、B的块A', B'利用CSR存储为$\mathbf{x}, \mathbf{x}^{(J)}, \mathbf{x}^{(R)}$和$\mathbf{y}, \mathbf{y}^{(J)}, \mathbf{y}^{(R)}$。

利用稠密矩阵行行划分的思想设计算法，将A'继续分为p个块$A'_j, j = 0, ..., p-1$，每个A'_j包含m列。又对A'_j采用CSR存储在$\mathbf{x}^{(j)}, \mathbf{x}^{(J,j)}, \mathbf{x}^{(R,j)}$中。

行行划分下的稀疏矩阵的并行乘法

算法9.20, 稀疏矩阵乘的行行划分并行算法

1 for k=0 to p-1 do $x_0^{(R,k)} = 1$;

2 for i=0 to m-1 do (得到A的子块的CSR存储)

  Set $k = 0$; $l = x_i^{(R,0)}$;

  for $j = x_i^{(R)}$ to $x_{i+1}^{(R)} - 1$ do

    if $x_j^{(J)} \geq km + m$ then

      Set $x_{i+1}^{(R,k)} = l$; $k = k + 1$; $l = x_i^{(R,k)}$;

    else

      $x_l^{(k)} = x_j$; $x_l^{(J,k)} = x_j^{(J)}$; $l = l + 1$;

    endif

  endfor

  Set $x_{i+1}^{(R,p-1)} = l$;

endfor

行行划分下的稀疏矩阵并行乘法

3 for j=0 to p-1 do

Compute $D' = A'_{(myrank+j)modp} \times B'$ with algorithm 9.19;

Compute $C' = C' + D'$ row by row with algorithm 9.17;

Send $\mathbf{y}, \mathbf{y}^{(J)}, \mathbf{y}^{(R)}$ to $P_{(myrank-1+p)modp}$;

Recv $\mathbf{y}, \mathbf{y}^{(J)}, \mathbf{y}^{(R)}$ from $P_{(myrank+1)modp}$;

endfor

练习

- 参照算法9.5编制矩阵乘法子程序。

Part II

线性方程组的并行求解

Outline

Outline

稠密矩阵的并行LU分解

考虑问题：

$$Ax = b$$

其中A是n阶方阵，b是右端项。求解稠密线性方程组的常用方法是LU分解法，即存在置换矩阵Q，使得$QA = LU$,其中L、U分别为下三角、上三角矩阵，于是方程变为：

$$QAx = LUx = Qb,$$

之后通过求解三角形方程组$Ly = Qb, Ux = y$得到原方程的解。

我们采用部分选主元的Gauss消去法进行列消元。在算法中A_k表示矩阵A的第k行。

算法10.1 稠密矩阵的串行LU分解算法

1. for j=0 to n-2 do
    Find l such that $|a_{lj}| = \max\{|a_{ij}|, i = j, ..., n-1\}$
    if $l \neq j$, swap A_j and A_l
    if $a_{jj} = 0$, A is singular and return
    $a_{ij} = a_{ij}/a_{jj}, i = j+1, ..., n-1$
    for k=j+1 to n-1 do
        $a_{ik} = a_{ik} - a_{ij} \times a_{jk}, i = j+1, ..., n-1$
    endfor
   endfor

稠密矩阵的并行LU分解算法

一维卷帘存储: A的第i列存放在$P_{i\ mod\ p}$中。例如:

P_0			P_1			P_2	
a_{00}	a_{03}	a_{06}	a_{01}	a_{04}	a_{07}	a_{02}	a_{05}
a_{10}	a_{13}	a_{16}	a_{11}	a_{14}	a_{17}	a_{12}	a_{15}
a_{20}	a_{23}	a_{26}	a_{21}	a_{24}	a_{27}	a_{22}	a_{25}
a_{30}	a_{33}	a_{36}	a_{31}	a_{34}	a_{37}	a_{32}	a_{35}
a_{40}	a_{43}	a_{46}	a_{41}	a_{44}	a_{47}	a_{42}	a_{45}
a_{50}	a_{53}	a_{56}	a_{51}	a_{54}	a_{57}	a_{52}	a_{55}
a_{60}	a_{63}	a_{66}	a_{61}	a_{64}	a_{67}	a_{62}	a_{65}
a_{70}	a_{73}	a_{76}	a_{71}	a_{74}	a_{77}	a_{72}	a_{75}

Table: 一个8阶矩阵在3个进程上的列循环划分

稠密矩阵的并行LU分解算法：算法描述

设$n = m \times p$, 我们将算法描述如下：

算法10.2 并行LU分解算法

1. icol = 0;
2. for j=0 to n-2 do
    - if myid = j mod p then (第j列对角元在本进程)
        - find l:$|a_{l,icol}| = \max\{|a_{i,icol}|, i = j, ..., n-1\}$
        - if $l \neq j$, swap A_j and A_l
        - if $a_{j,icol} = 0$, A is singular and return;
        - $a_{i,icol} = a_{i,icol}/a_{j,icol}, i = j+1, ..., n-1$
        - $f_{i-j-1} = a_{i,icol}, i = j+1, ..., n-1$
        - Send l and f to P_{myid+1}
        - icol = icol + 1

稠密矩阵的并行LU分解算法：算法描述(续)

2 (cont) else

Recv l and f from P_{myid-1}

if $myid + 1 \neq j$ mod p then

send l and f to P_{myid+1}

endif

endif

if $l \neq j$, swap A_j nad A_l

for k=icol to m-1 do

$a_{ik} = a_{ik} - f_i \times a_{jk}, i = j+1, ..., n-1$

endfor

endfor

Outline

三角方程组的并行解法

考虑下三角方程组$Lx = b$，其串行算法为：

算法10.3 下三角线性方程组的串行解法

1. for i=0 to n-1 do
   $x_i = b_i / l_{ii}$;
   for j=i+1 to n-1 do
   $b_j = b_j - l_{ji} \times x_i$;
   endfor
   endfor

每次对**b**修正的时候用到L的一列，称为列扫描算法，可直接并行化：L按行分块使得计算并行，卷帘存储使得负载均衡。
缺点：每次算出x_i后要广播，通信次数多，每次的计算量又不大。
适合 共享存储计算机。

三角方程组的并行解法(续)

数据存储结构：

P_0			P_1			P_2	
l_{00}							
l_{10}			l_{11}				
l_{20}			l_{21}			l_{22}	
l_{30}	l_{33}		l_{31}			l_{32}	
l_{40}	l_{43}		l_{41}	l_{44}		l_{42}	
l_{50}	l_{53}		l_{51}	l_{54}		l_{52}	l_{55}
l_{60}	l_{63}	l_{66}	l_{61}	l_{64}		l_{62}	l_{65}
l_{70}	l_{73}	l_{76}	l_{71}	l_{74}	l_{77}	l_{72}	l_{75}

Table: 一个8阶下三角矩阵在3个进程上的列循环划分

算法10.4 下三角线性方程组的列循环划分并行解法（非阻塞）

1. k=0
2. if myid = 0, then
   $u_i = b_i, i = 0, 1, ..., n-1,\ v_i = 0, i = 0, 1, ..., p-2$
   else
   $u_i = 0, i = 0, 1, ..., n-1$
3. for i=myid step p to n-1 do
   if $i > 0$, recv v from $P_{(i-1)modp}$
   $x_k = (u_i + v_0)/(l_{ik})$
   $v_j = v_{j+1} + u_{i+1+j} - l_{i+1+j,k} \times x_k, j = 0, 1, ..., p-3$
   $v_{p-2} = u_{i+p-1} - l_{i+p-1,k} \times x_k$
   Send v to $P_{(i+1)modp}$
   $u_j = u_j - l_{jk} \times x_k, j = i+p, ..., n-1$
   k=k+1;
   endfor

Outline

三对角方程的并行求解算法

三对角方程组一个来源：一维椭圆方程中心差分格式离散。系数矩阵A具有三对角形状，即非零元只分布在主对角线和上下两条次对角线上，其余元素为零，下图为一个$n=12$的三对角矩阵：

$$
\left[\begin{array}{cccc|cccc|cccc}
a_1 & b_1 & & & & & & & & & & \\
c_2 & a_2 & b_2 & & & & & & & & & \\
 & c_3 & a_3 & b_3 & & & & & & & & \\
 & & c_4 & a_4 & b_4 & & & & & & & \\
\hline
 & & & c_5 & a_5 & b_5 & & & & & & \\
 & & & & c_6 & a_6 & b_6 & & & & & \\
 & & & & & c_7 & a_7 & b_7 & & & & \\
 & & & & & & c_8 & a_8 & b_8 & & & \\
\hline
 & & & & & & & c_9 & a_9 & b_9 & & \\
 & & & & & & & & c_{10} & a_{10} & b_{10} & \\
 & & & & & & & & & c_{11} & a_{11} & b_{11} \\
 & & & & & & & & & & c_{12} & a_{12}
\end{array}\right]
$$

对三角线方程组$A\mathbf{x}=\mathbf{r}$，最常用的串行算法是追赶法，其算法如下：
算法10.5， 追赶法

1. for i=2 to n do
   $c_i = c_i/a_{i-1}; a_i = a_i - c_i b_{i-1};$
   endfor
2. $x_1 = r_1;$
   for i=2 to n do $x_i = r_i - c_i x_{i-1};$
3. $x_n = x_n/a_n;$
   for i=n-1 to 1 do $x_i = (x_i - b_i x_{i+1})/a_i$.

该算法总计算量约为$O(8n)$,每个后续操作严格依赖于前一个操作，故不可能直接进行并行化，进行并行计算的时候需要对算法进行重新设计。

三对角线方程组的并行求解（续）

矩阵分裂法

假设用p个进程并行求解三对角方程组，设n可被p整除，记$m = n/p$，将矩阵分成p块，即把$c_j, a_j, b_j, (j = im + 1, ..., im + m)$存储在进程$P_i(i = 0, 1, ..., p - 1)$上，对右端项与解也类似存储。分裂法分为三个步骤:

- 局部消元;
- 解耦;
- 回代求解。

局部消元:

$$
\left(\begin{array}{cccc|cccc|cccc}
a_1 & & & & b_1 & & & & & & & \\
& a_2 & & & b_2 & & & & & & & \\
& & a_3 & & b_3 & & & & & & & \\
& & & a_4 & b_4 & & & & & & & \\
\hline
& & & c_5 & a_5 & & & & b_5 & & & \\
& & & c_6 & & a_6 & & & b_6 & & & \\
& & & c_7 & & & a_7 & & b_7 & & & \\
& & & c_8 & & & & a_8 & b_8 & & & \\
\hline
& & & & & & & c_9 & a_9 & & & \\
& & & & & & & c_{10} & & a_{10} & & \\
& & & & & & & c_{11} & & & a_{11} & \\
& & & & & & & c_{12} & & & & a_{12}
\end{array}\right)
$$

三对角线方程组的并行求解（续）

解耦：

局部消元后，进程P_0上最后一个方程、 P_{p-1}上第一个方程、其它进程上的第一个与最后一个方程为：

$$
\begin{aligned}
a_m x_m + b_m x_{m+1} &= r_m, \\
c_{(p-1)m+1} x_{(p-1)m} + a_{(p-1)m+1} x_{(p-1)m+1} &= r_{(p-1)m+1}, \\
c_{im+1} x_{im} + a_{im+1} x_{im+1} + b_{im+1} x_{(i+1)m+1} &= r_{im+1}, \\
c_{im+m} x_{im} + a_{im+m} x_{im+m} + b_{im+m} x_{(i+1)m+1} &= r_{im+m}.
\end{aligned}
$$

这构成了一个新的三对角方程组

$$
\begin{pmatrix}
d_0 & e_0 & & \\
f_1 & d_1 & \ddots & \\
 & \ddots & \ddots & e_{2p-4} \\
 & & f_{2p-3} & d_{2p-3}
\end{pmatrix}
\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{2p-3} \end{pmatrix}
=
\begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{2p-3} \end{pmatrix}
$$

三对角线方程组的并行求解（续）

其中

- $f_{2i} = c_{im+m}, d_{2i} = b_{im+m}, e_{2i} = a_{im+m}$,
- $f_{2i+1} = a_{im+m+1}, d_{2i+1} = c_{im+m+1}, e_{2i+1} = b_{im+m+1}$,
- $y_{2i} = x_{im+m+1}, y_{im+m}, s_{2i} = r_{im+m}, s_{2i+1} = r_{im+m+1}$.

解耦三对角方程规模：2p-2阶，相对较小，可以直接用追赶法（算法10.5）求解。
在上述方程组求解之后，可利用其解回代计算其它x的分量。

算法10.6 三对角方程组的并行分裂法

1. for i=2 to m do
   $t = c_i/a_{i-1}; c_i = -tc_{i-1}; a_i = a_i - tb_{i-1}; r_i = r_i - tr_{i-1};$
   endfor
2. for i=m-1 to 1 do
   $t = b_i/a_{i+1}; b_i = -tb_{i+1}; c_i = c_i - tc_{i+1}; r_i = r_i - tr_{i+1};$
   endfor
3. Gather $c_{im+k}, a_{im+k}, b_{im+k}, r_{im+k}$, for k=1 and m to P_0;
   if myid = 0 then solve the decouple equation.
   Scatter y from P_0 to other processes;
4. if myid = 0 then
   for i=1 to m-1 do $x_i = (r_i - b_i y_0)/a_i$;
   else if myid = p-1 then
   for i=2 to m do $x_i = (r_i - c_i y_{2p-3})/a_i$;
   else
   for i=2 to m-1 do $x_i = (r_i - c_i y_{myid\times 2-1} - b_i y_{myid\times 2})$;
   endif

带状线性方程组的并行求解

带状线性方程组：$Ax = r$，其中A除对角线外，在上、下三角部分分别在另外β, α条紧靠对角线并与之平行的直线上元素不为零， 其它位置元素都为零的矩阵。

$$\begin{bmatrix} A_1 & B_1 & & \\ C_2 & A_2 & B_2 & \\ & C_3 & A_3 & B_3 \\ & & C_4 & A_4 & B_4 \end{bmatrix}.$$

借鉴三对角方程的算法，这类方程的并行分裂法仍可以分为三步：

- 局部消元；
- 解耦；
- 回代；

带状线性方程组的并行求解

- 解耦方程的构成：
  - 第一个进程的后α个方程
  - 最后一个进程的前β个方程
  - 其余进程的后α个方程和前β个方程

  即：前p-1个进程各自的最后α个方程，后$p-1$个进程各自的前β个方程。
- 解耦方程的规模：

$$(p-1)(\alpha+\beta)$$

解耦方程仍然是一个带状线性（分块三对角）方程组，规模相对较小，可在单个进程直接求解。

Outline

Jacobi迭代法

对方程$A\mathbf{x}=\mathbf{b}$，A是$n\times n$矩阵，记$D,-L,-U$分别是其对角线、严格上三角、严格下三角构成的矩阵，Jacobi迭代是指如下形式的迭代：

$$\mathbf{x}^{k+1}=D^{-1}(L+U)\mathbf{x}^k+D^{-1}\mathbf{b},$$

第k+1次迭代的每个分量只依赖于第k次迭代时的分量，故只要第k次迭代的分量已经传送到相应的进程上，第k+1次迭代时各分量就可以并行进行。
Jacobi迭代的并行算法我们已经在之前实现，根据不同的矩阵和向量分块，可以很容易地确定通信结构。

Gauss-Seidel迭代

串行Jacobi迭代(分量形式):

1. for i=1:n
   $x_i^{k+1} = (b_i - \sum_{j=1}^{i-1} a_{ij} x_j^k - \sum_{j=i+1}^{n} a_{ij} x_j^k)/a_{ii}$
   end

Gauss-Seidel迭代法是逐个分量进行计算的一种方法,

1. for i=1:n
   $x_i^{k+1} = (b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} x_j^k)/a_{ii}$
   end

从矩阵分裂的角度Gauss-Sediel迭代可以表示为:

$$(D-L)\mathbf{x}^{k+1} = U\mathbf{x}^k + b$$

从迭代过程来看，每计算一个新的分量都需要前面所有新计算出来的分量的结果，这是一个严格的串行过程。但也可以设计出并行算法：
设$s_i = \sum_{j=i+1}^{n} a_{ij} x_j^0, i = 1, 2, ..., n-1, s_n = 0$。则并行算法如下：

算法10.7 并行Gauss-Seidel迭代算法

1. k =0
2. for i=1 to n do
   $x_i^{k+1} = (b_i - s_i)/a_{ii}, a_i = 0;$
   for j=1 to n, $j \neq i$ do
   $s_j = s_j + a_{ji} x_i^{k+1}$
   endfor
   endfor
3. $\|\mathbf{x}^{k+1} - \mathbf{x}^k\|_2 \le \epsilon \|\mathbf{x}^{k+1} - \mathbf{x}^0\|_2$ 停止， 否则$k = k + 1$.

每次s_j是可以并行计算的，截止条件的计算也可以并行。

Outline

共轭梯度法(CG):

1. k=0, $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$
2. while $\mathbf{r}_k \neq 0$
   k=k+1;
   if k=1
   $\mathbf{p}_1 = \mathbf{p}_0$;
   else
   $\beta_k = (\mathbf{r}_{k-1}, \mathbf{r}_{k-1})/(\mathbf{r}_{k-2}, \mathbf{r}_{k-2})$
   $\mathbf{p}_k = \mathbf{r}_{k-1} + \beta_k \mathbf{p}_{k-1}$
   endif
   $\alpha_k = (\mathbf{r}_{k-1}, \mathbf{r}_{k-1})/(A\mathbf{p}_k, \mathbf{p}_k)$
   $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_k \mathbf{p}_k$
   $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_k A\mathbf{p}_k$
   end
3. $\mathbf{x} = \mathbf{x}_k$

广义极小残量(GMRES)法:

1. $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}, h_{10} = \|\mathbf{r}_0\|_2$,k=0
2. while $(h_{k+1,k} > 0)$

   $\mathbf{q}_{k+1} = \mathbf{r}_k / h_{k+1,k}$
   k=k+1
   $\mathbf{r}_k = A\mathbf{q}_k$
   for i=1:k
   $h_{ik} = (\mathbf{q}_i, \mathbf{r}_k)$
   $\mathbf{r}_k = \mathbf{r}_k - h_{ik}\mathbf{q}_i$
   end
   $h_{k+1,k} = \|\mathbf{r}_k\|_2$
   $\mathbf{x}_k = x_0 + Q_k\mathbf{y}_k$ where $\|h_{10}\mathbf{e}_1 - \tilde{H}_k\mathbf{y}_k\|_2 = \min$

   end
3. $\mathbf{x} = \mathbf{x}_k$

这里$\tilde{H}_k$为上Hessenberg矩阵。

具体算法的推导及更多Krylov子空间迭代法，可以参看 G. H. Golub and C. F. Van Loan的“Matrix Computations”或 Y. Saad的”Iterative Methods for Sparse Linear System”。

这类算法所涉及的运算：

- 矩阵向量乘
- 向量相加
- 向量乘标量
- 向量内积
- 低维上Hessenberg方程组或低维最小二乘问题的求解。

算法设计策略

- 对所有n维向量，全部采用相同的块分布方式进行存储
- 对系数矩阵，采用适当的方式存储，以便于进行矩阵向量乘
- 向量相加及向量乘标量可以完全并行地进行，没有任何通信。
- 低维小线性方程组与低维最小二乘问题的求解的计算量很小，可以重复计算。
- 向量内积可以先局部计算，再归约，若需要，可以进一步广播。

其它问题

- 异步并行迭代算法：基于压缩映像原理的迭代法，算法不需要处理机之间互相等待，能充分发挥并行机的工作效率，缺点是收敛速度慢。
- 代数特征值问题的并行：基于矩阵向量乘和向量内积。
- Krylov子空间迭代法的预条件：关键是提出有效的预条件矩阵，并且关于预条件矩阵的线性方程组能快速求解。
- 调用成熟软件包：与自己开发相比，这些软件包比较成熟，高效，缺点是不如自己开发用起来更为容易上手。

比较流行的软件包

- 稠密线性方程组的求解：
  LINPACK、LAPACK、ScaLAPACK ...;
- 一般稀疏非对称线性方程组：
  SuperLU（SuperLU_DIST）、MUMPS ... ；
- 一般稀疏线性方程组并行迭代法：
  SPOOLES、Trilions(AztecOO)、PETSc ...;
- 并行预条件软件包：
  Hypre...
- 稀疏矩阵特征值： SLEPC

PETSc

PETSc(Portable, Extensible Tookit for Scientific Computation) 1991.9 开始研制， Argonne国家实验室开发。 基于BLAS、LAPACK和MPI，包含三个基本组件:

- SLES(Linear Solver)
- SNES(Nonlinear Solver)
- TS (Time Stepping)

提供丰富的Krylov子空间迭代法和预条件子。 下载地址:

`http://www.mcs.anl.gov/petsc/`

Hypre

Hypre:High Performance Preconditioners, Lawrence Livermore National Lab(LLNL)开发，主要用于大规模并行机上求解大型稀疏线性方程组。

主要特点:

- 可扩展的预条件子：包含可扩展求解超大规模稀疏线性方程组的几类预条件子，比如代数多重网格
- 常用的迭代算法：Krylov子空间迭代法，如GMRES、CG（PCG，CGNR，BiCGStab）

下载地址:

`https://computation.llnl.gov/casc/hypre/software.html`

安装方便，大多数情况下，直接运行configure后make编译即可。

作业

考虑第六次课后面用5点差分格式离散后的方程，如果用CSR格式或是Aij格式存储离散得到的系数矩阵，写一个矩阵形式的串行Jacobi迭代程序， 如果有可能把它并行化。