线性方程组并行直接法

(基于 MPI)

—— 选主元 LU 分解

—— 三角矩阵求解

线性方程组求解

Linear algebra — in particular, **the solution of linear systems of equations** — lies at the **heart** of most calculations in scientific computing.

— Dongarra & Eijkhout

J. J. Dongarra and V. Eijkhout, Numerical linear algebra algorithms and software, *JCAM*, 123 (2000), 489–514

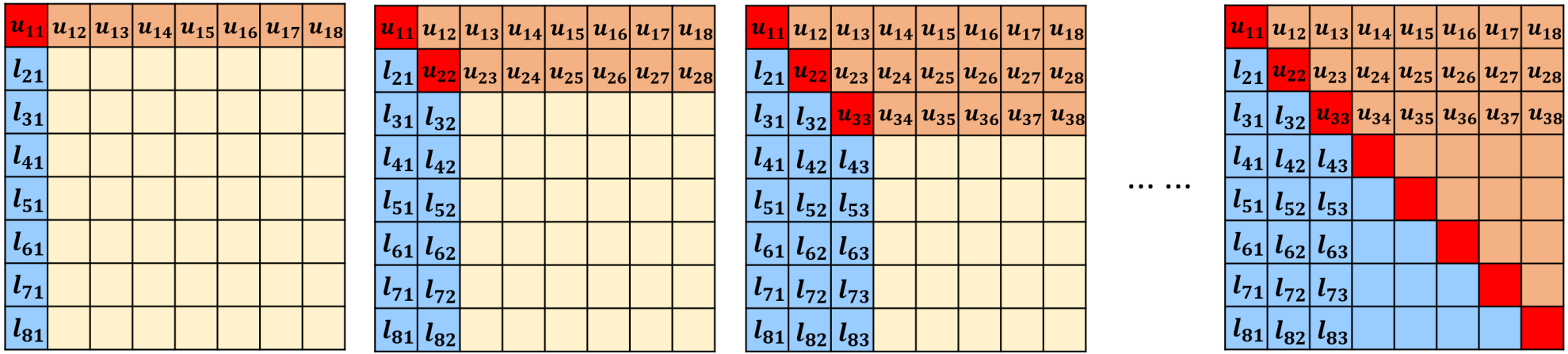
- 线性方程组是许多重要问题的核心，有效地求解线性方程组在科学与工程计算中非常重要
- 并行计算机的问世，使求解问题的速度和规模大幅提高，同时也使计算方法产生了变化
- 软件包：**Lapack**，**ScaLapack**

LU 分解

$$PA = LU$$

数据存储方式

- LU 分解的主要计算量是更新矩阵 A。
- 根据算法计算过程可知，如果是按列（或按行）连续分块存储在各个结点，则会出现越往后计算越多结点空闲的情况，因此建议采用卷帘方式存储。



数据存储方式

- 可以采用一维划分，也可以采用二维划分。在实际应用中，通常采用二维划分，即在两个方向上都进行循环划分，然后存储到按二维排列的结点上。

A_0	A_1	A_2

A_{00}	A_{01}	A_{02}
A_{10}	A_{11}	A_{12}
A_{20}	A_{21}	A_{22}

▶ 为简单起见，这里介绍一维列循环划分的并行算法

串行算法

算法 1.8 部分选主元 LU 分解

1: $p = 1 : n$ % 用于记录置换矩阵
2: **for** $k = 1$ to $n - 1$ **do**
3: $[a_{\max}, l] = \max_{k \le i \le n} |a_{ik}|$ % 选列主元, 其中 l 表示主元所在的行
4: **if** $l \neq k$ **then**
5: **for** $j = 1$ to n **do**
6: $tmp = a_{kj}, a_{kj} = a_{lj}, a_{lj} = tmp$ % 交换第 k 行与第 l 行
7: **end for**
8: $tmp = p(k), p(k) = p(l), p(l) = tmp$ % 更新置换矩阵
9: **end if**
10: **for** $i = k + 1$ to n **do**
11: $a_{ik} = a_{ik}/a_{kk}$ % 计算 L 的第 k 列
12: **end for**
13: **for** $i = k + 1$ to n **do**
14: **for** $j = k + 1$ to n **do**
15: $a_{ij} = a_{ij} - a_{ik} * a_{kj}$ % 更新 $A(k+1:n, k+1:n)$
16: **end for**
17: **end for**
18: **end for**

并行算法

计算过程

- 每次选主元时，只涉及到同一列，因此没有数据通信
- 确定主元后，计算 L 的第 k 列
- 将 l（主元所在的行）和 L 的第 k 列传给其他结点
- 各处理器（进程）更新自己的数据

并行算法

1: $icol = 0$
2: **for** $j = 0$ to $n-2$ **do**
3: **if** $myid = j \mod p$ **then**
4: find l : $|a_{l,icol}| = \max\{|a_{i,icol}|,\ i = j, j+1, \ldots, n-1\}$
5: **if** $a_{l,icol} = 0$ **then** kill all processes % A is singular and return
6: **if** $l \neq j$ **then** swap $a_{j,icol}$ and $a_{l,icol}$
7: $a_{i,icol} = a_{i,icol}/a_{j,icol}, i = j+1, \ldots, n-1$
8: $f_{i-j-1} = a_{i,icol}, i = j+1, \ldots, n-1$
9: send$(l, myid+1 \mod p)$ and send$(f, myid+1 \mod p)$
10: $icol = icol + 1$
11: **else**
12: recv$(l, myid-1 \mod p)$ and recv$(f, myid-1 \mod p)$ % 广播 l 和 f
13: **if** $myid+1 \neq j \mod p$ **then** send$(l, myid+1 \mod p)$ and send$(f, myid+1 \mod p)$
14: **end if**
15: **if** $l \neq j$ **then** swap $A_{j,:}$ and $A_{l,:}$
16: **for** $k = icol$ to $m-1$ **do**
17: $a_{ik} = a_{ik} - f_{i-j-1}a_{jk}, i = j+1, \ldots, n-1$
18: **end for**
19: **end for**

华东师范大学 数学科学学院
School of Mathematical Sciences, ECNU

三角方程并行求解

—— 以 $Lx = b$ 为例

三角方程组

$$\begin{bmatrix} l_{00} & & & \\ l_{10} & l_{11} & & \\ \vdots & & \ddots & \\ l_{n-1,0} & l_{n-1,1} & \cdots & l_{n-1,n-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{bmatrix}$$

列扫描算法

$$x_0 = b_0 / l_{00}$$

$$x_1 = \left(b_1 - l_{10}x_0\right) / l_{11}$$

$$x_2 = \left(b_2 - l_{20}x_0 - l_{21}x_1\right) / l_{22}$$

$$x_3 = \left(b_3 - l_{30}x_0 - l_{31}x_1 - l_{32}x_2\right) / l_{33}$$

$$\vdots$$

$$x_i = \left(b_i - l_{i0}x_0 - l_{i1}x_1 - \cdots - l_{i,i-1}x_{i-1}\right) / l_{ii}$$

$$\vdots$$

并行算法

- 基于列扫描的共享内存并行算法
  - 串行计算 x_0，然后所有线程并行更新 $b_i = b_i - l_{i0}x_0,\ i = 1,2,\ldots,n-1$
  - 串行计算 x_1，然后所有线程并行更新 $b_i = b_i - l_{i1}x_1,\ i = 2,3,\ldots,n-1$
  - 串行计算 x_2，然后所有线程并行更新 $b_i = b_i - l_{i2}x_2,\ i = 3,4,\ldots,n-1$
  - ……

- 基于列扫描的分布式内存并行算法
  - 若数据按行存放，则可以采用上面的计算方案
  - 若数据按列存在，则需要修改计算方案，改为采用**流水线技术**

流水线计算方法

几点说明

- 矩阵 A 按列采用卷帘方式存储在各个处理器（进程）中
- 进程间每次传递的是部分修正后的右端项，而不是新求出的解，通过叠加的方式计算下次的解

并行算法

1: $k = 0$
2: **if** $myid = 0$ **then**
3: $\quad u_i = b_i,\ i = 0, 1, \ldots, n-1$
4: $\quad v_i = 0,\ i = 0, 1, \ldots, p-2$
5: **else**
6: $\quad u_i = 0,\ i = 0, 1, \ldots, n-1$
7: **end if**
8: **for** $i = myid$ to $n-1$ with stepsize p **do** % 即 $i = myid : p : n-1$
9: $\quad$ **if** $i > 0$ **then** recv$(v, i-1 \mod p)$
10: $\quad x_k = (u_i + v_0)/l_{ik}$
11: $\quad v_j = v_{j+1} + u_{i+j+1} - l_{i+j+1,k}x_k,\ j = 0, 1, \ldots, p-3$
12: $\quad v_{p-2} = u_{i+p-1} - l_{i+p-1,k}x_k$
13: $\quad$ send$(v, i+1 \mod p)$
14: $\quad u_j = u_j - l_{jk}x_k,\ j = i+p, \ldots, n-1$
15: $\quad k = k+1$
16: **end for**

以 p=3 为例

- 带波浪号表示本地数据

P_0	P_1	P_2
$k=0,\ u=b,\ v=0,\ i=0$ $\tilde{x}_0 = u_0/\tilde{l}_{00} = b_0/l_{00} = x_0$ $v_0 = u_1 - \tilde{l}_{10}\tilde{x}_0 = b_1 - l_{10}x_0$ $v_1 = u_2 - \tilde{l}_{20}\tilde{x}_0 = b_2 - l_{20}x_0$	$k=0,\ u=0,\ i=1$	$k=0,\ u=0,\ i=2$
$send(v, P_1)$ $u_j = u_j - \tilde{l}_{j0}\tilde{x}_0 = b_j - l_{j0}x_0$ $j = 3, 4, \ldots, n-1$	$recv(v, P_0)$ $\tilde{x}_0 = (u_0 + v_0)/\tilde{l}_{10}$ $= (b_1 - l_{10}x_0)/l_{11} = x_1$ $v_0 = v_1 + u_2 - \tilde{l}_{20}\tilde{x}_0$ $= b_2 - l_{20}x_0 - l_{21}x_1$ $v_1 = u_3 - \tilde{l}_{30}\tilde{x}_0 = -l_{31}x_1$	
$k = k+1 = 1$ $i = i+p = 3$	$send(v, P_2)$ $u_j = u_j - \tilde{l}_{j0}\tilde{x}_0 = -l_{j1}x_1$ $j = 4, 5, \ldots, n-1$	$recv(v, P_1)$ $\tilde{x}_0 = (u_0 + v_0)/\tilde{l}_{20}$ $= (b_2 - l_{20}x_0 - l_{21}x_1)/l_{22} = x_2$ $v_0 = v_1 + u_3 - \tilde{l}_{30}\tilde{x}_0$ $= -l_{31}x_1 - l_{32}x_2$ $v_1 = u_3 - \tilde{l}_{40}\tilde{x}_0 = -l_{42}x_2$
$recv(v, P_2)$ $\tilde{x}_1 = (u_3 + v_0)/\tilde{l}_{31}$ $= (b_3 - l_{30}x_0 - l_{31}x_0$ $= -l_{32}x_2)/l_{33} = x_3$ $v_0 = v_1 + u_4 - \tilde{l}_{41}\tilde{x}_1$ $= b_4 - l_{40}x_0 - l_{42}x_2 - l_{43}x_3$ $v_1 = u_5 - \tilde{l}_{41}\tilde{x}_1$ $= b_5 - l_{50}x_0 - l_{54}x_4$	$k = k+1 = 1$ $i = i+p = 4$	$send(v, P_0)$ $u_j = u_j - \tilde{l}_{j0}\tilde{x}_0 = -l_{j2}x_2$ $j = 5, 6, \ldots, n-1$
$send(v, P_1)$ $\vdots$	$recv(v, P_0)$ $\vdots$	$\vdots$

谢 谢
THANK YOU