ALGORITHMES NUMERIQUES PARALLELES : vecteurs et matrices

Daniel Etiemble
LRI, Université Paris Sud
de@lri.fr

Références

- M. T. Heath, « Parallel Numerical Algorithms : vector and matrix products », University of Illinois at Urbana Champaign, CS 454
- Parallel Matrix-Matrix Multiplication, www.cs.indiana.edu/classes/b673/notes/matrix_mult.html

Introduction

- Exemples de calculs sur vecteurs et matrices
  - Produit scalaire
  - Produit « externe »
  - Produit matrice-vecteur
  - Produit de matrices
- Décomposition de domaines
  - 1 dimension : lignes ou colonnes
  - 2 dimensions
- Réduire le ratio communications/calculs
- Minimiser l'occupation mémoire

Produit scalaire

- Le produit scalaire de deux vecteurs x[n] et y[n] est

$$x^T y = \sum_{i=1}^{n} x_i y_i$$

- Parallélisation

$$\boxed{x_1 y_1 + x_2 y_2} + \boxed{x_3 y_3 + x_4 y_4} + \boxed{x_5 y_5 + x_6 y_6} + \boxed{x_7 y_7 + x_8 y_8}$$

+

- Produit scalaire local + réduction

Produit vecteur-vecteur transposé (« outer product »)

- Le produit externe $x.y^T$ de deux vecteurs x[n] et y[n] est une matrice [n][n] dont l'entrée (i,j) est $x_i y_j$
- Exemple
- Temps calcul série
  - $T_1 = n^2 t_c$
- Stratégies de parallélisation avec p processeurs
  - Décomposition 1 dimension
    - Lignes ou colonnes
  - Décomposition 2 dimensions
    - Sous blocs

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}^T = \begin{bmatrix} x_1 y_1 & x_1 y_2 & x_1 y_3 \\ x_2 y_1 & x_2 y_2 & x_2 y_3 \\ x_3 y_1 & x_3 y_1 & x_3 y_1 \end{bmatrix}$$

$$\begin{bmatrix} x_1y_1 & x_1y_2 & x_1y_3 & x_1y_4 & x_1y_5 & x_1y_6 & x_1y_7 & x_1y_8 \\ x_2y_1 & x_2y_2 & x_2y_3 & x_2y_4 & x_2y_5 & x_2y_6 & x_2y_7 & x_2y_8 \\ x_3y_1 & x_3y_2 & x_3y_3 & x_3y_4 & x_3y_5 & x_3y_6 & x_3y_7 & x_3y_8 \\ x_4y_1 & x_4y_2 & x_4y_3 & x_4y_4 & x_4y_5 & x_4y_6 & x_4y_7 & x_4y_8 \\ x_5y_1 & x_5y_2 & x_5y_3 & x_5y_4 & x_5y_5 & x_5y_6 & x_5y_7 & x_5y_8 \\ x_6y_1 & x_6y_2 & x_6y_3 & x_6y_4 & x_6y_5 & x_6y_6 & x_6y_7 & x_6y_8 \\ x_7y_1 & x_7y_2 & x_7y_3 & x_7y_4 & x_7y_5 & x_7y_6 & x_7y_7 & x_7y_8 \\ x_8y_1 & x_8y_2 & x_8y_3 & x_8y_4 & x_8y_5 & x_8y_6 & x_8y_7 & x_8y_8 \end{bmatrix}$$

Produit « externe » : décomposition 1D

$$\begin{array}{c} \text{P0} \\ \\ \text{P1} \\ \\ \text{P2} \\ \\ \text{P3} \\ \\ \end{array} \begin{bmatrix} x_1y_1 & x_1y_2 & x_1y_3 & x_1y_4 & x_1y_5 & x_1y_6 & x_1y_7 & x_1y_8 \\ x_2y_1 & x_2y_2 & x_2y_3 & x_2y_4 & x_2y_5 & x_2y_6 & x_2y_7 & x_2y_8 \\ x_3y_1 & x_3y_2 & x_3y_3 & x_3y_4 & x_3y_5 & x_3y_6 & x_3y_7 & x_3y_8 \\ x_4y_1 & x_4y_2 & x_4y_3 & x_4y_4 & x_4y_5 & x_4y_6 & x_4y_7 & x_4y_8 \\ x_5y_1 & x_5y_2 & x_5y_3 & x_5y_4 & x_5y_5 & x_5y_6 & x_5y_7 & x_5y_8 \\ x_6y_1 & x_6y_2 & x_6y_3 & x_6y_4 & x_6y_5 & x_6y_6 & x_6y_7 & x_6y_8 \\ x_7y_1 & x_7y_2 & x_7y_3 & x_7y_4 & x_7y_5 & x_7y_6 & x_7y_7 & x_7y_8 \\ x_8y_1 & x_8y_2 & x_8y_3 & x_8y_4 & x_8y_5 & x_8y_6 & x_8y_7 & x_8y_8 \end{bmatrix}$$

Décomposition 1D par ligne
  - n/p lignes de n éléments
  - Si x et y sont stockés dans le processeur 0
    - Diffusion des n éléments de y aux (p-1) autres processeurs
    - Envoi de n/p éléments de x à chacun des (p-1) autres processeurs
  - Chaque processeur calcule n^2/p produits
- Décomposition 1 D par colonnes
  - Idem (permutation du rôle des lignes et colonnes)

Produit « externe » : décomposition 2D

$$
\begin{array}{c}\text{P0}\\[5em]\text{P1}\end{array}
\begin{bmatrix}
x_1y_1 & x_1y_2 & x_1y_3 & x_1y_4 & x_1y_5 & x_1y_6 & x_1y_7 & x_1y_8 \\
x_2y_1 & x_2y_2 & x_2y_3 & x_2y_4 & x_2y_5 & x_2y_6 & x_2y_7 & x_2y_8 \\
x_3y_1 & x_3y_2 & x_3y_3 & x_3y_4 & x_3y_5 & x_3y_6 & x_3y_7 & x_3y_8 \\
x_4y_1 & x_4y_2 & x_4y_3 & x_4y_4 & x_4y_5 & x_4y_6 & x_4y_7 & x_4y_8 \\
x_5y_1 & x_5y_2 & x_5y_3 & x_5y_4 & x_5y_5 & x_5y_6 & x_5y_7 & x_5y_8 \\
x_6y_1 & x_6y_2 & x_6y_3 & x_6y_4 & x_6y_5 & x_6y_6 & x_6y_7 & x_6y_8 \\
x_7y_1 & x_7y_2 & x_7y_3 & x_7y_4 & x_7y_5 & x_7y_6 & x_7y_7 & x_7y_8 \\
x_8y_1 & x_8y_2 & x_8y_3 & x_8y_4 & x_8y_5 & x_8y_6 & x_8y_7 & x_8y_8
\end{bmatrix}
\begin{array}{c}\text{P2}\\[5em]\text{P3}\end{array}
$$

- Décomposition en blocs de $n/p^{1/2}$ par $n/p^{1/2}$ éléments.
- Topologie virtuelle de $p^{1/2}$ lignes et $p^{1/2}$ colonnes
- Si x et y sont stockés dans le processeur 0
  - $p^{1/2}$ diffusions de $n/p^{1/2}$ éléments de x à tous les processeurs de chaque ligne de processeurs
  - $p^{1/2}$ diffusions de $n/p^{1/2}$ éléments de y à tous les processeurs de chaque colonnes de processeurs
- Chaque processeur calcule n^2/p produits

Produit matrice-vecteur

- Soit le produit matrice vecteur y = A x où A est une matrice (n,n) et x et y sont des vecteurs de n éléments
- Les éléments du vecteur y sont donnés par $y_i = \sum_{j=1}^{n} a_{i,j} x_j$

$$
\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix}
=
\begin{array}{c}
\begin{matrix} & + & & + & & + & & + & & + & & + & & + & \end{matrix} \\
\begin{bmatrix}
a_{11}x_1 & a_{12}x_2 & a_{12}x_3 & a_{12}x_4 & a_{12}x_5 & a_{12}x_6 & a_{12}x_7 & a_{12}x_8 \\
a_{21}x_1 & a_{22}x_2 & a_{23}x_3 & a_{24}y_4 & a_{25}x_5 & a_{26}x_6 & a_{27}x_7 & a_{28}x_8 \\
a_{31}x_1 & a_{32}x_2 & a_{33}x_3 & a_{34}x_4 & a_{35}x_5 & a_{36}y_6 & a_{37}x_7 & a_{38}x_8 \\
a_{41}x_1 & a_{42}x_2 & a_{43}x_3 & a_{44}x_4 & a_{45}x_5 & a_{46}x_6 & a_{47}x_7 & a_{48}x_8 \\
a_{51}x_1 & a_{52}x_2 & a_{53}x_3 & a_{54}x_4 & a_{55}x_5 & a_{56}x_6 & a_{57}x_7 & a_{58}x_8 \\
a_{61}x_1 & a_{62}x_2 & a_{63}x_3 & a_{64}x_4 & a_{65}x_5 & a_{66}x_6 & a_{67}y_7 & a_{68}x_8 \\
a_{71}x_1 & a_{72}x_2 & a_{73}x_3 & a_{74}x_4 & a_{75}x_5 & a_{76}x_6 & a_{77}x_7 & a_{78}x_8 \\
a_{81}x_1 & a_{82}x_2 & a_{83}x_3 & a_{84}x_4 & a_{85}x_5 & a_{86}x_6 & a_{87}x_7 & a_{88}x_8
\end{bmatrix}
\end{array}
$$

Produit matrice-vecteur – Décomposition 1D ligne

$$\begin{bmatrix} a_{11}x_1 & a_{12}x_2 & a_{12}x_3 & a_{12}x_4 & a_{12}x_5 & a_{12}x_6 & a_{12}x_7 & a_{12}x_8 \\ a_{21}x_1 & a_{22}x_2 & a_{23}x_3 & a_{24}y_4 & a_{25}x_5 & a_{26}x_6 & a_{27}x_7 & a_{28}x_8 \\ a_{31}x_1 & a_{32}x_2 & a_{33}x_3 & a_{34}x_4 & a_{35}x_5 & a_{36}y_6 & a_{37}x_7 & a_{38}x_8 \\ a_{41}x_1 & a_{42}x_2 & a_{43}x_3 & a_{44}x_4 & a_{45}x_5 & a_{46}x_6 & a_{47}x_7 & a_{48}x_8 \\ a_{51}x_1 & a_{52}x_2 & a_{53}x_3 & a_{54}x_4 & a_{55}x_5 & a_{56}x_6 & a_{57}x_7 & a_{58}x_8 \\ a_{61}x_1 & a_{62}x_2 & a_{63}x_3 & a_{64}x_4 & a_{65}x_5 & a_{66}x_6 & a_{67}x_7 & a_{68}x_8 \\ a_{71}x_1 & a_{72}x_2 & a_{73}x_3 & a_{74}x_4 & a_{75}x_5 & a_{76}x_6 & a_{77}x_7 & a_{78}x_8 \\ a_{81}x_1 & a_{82}x_2 & a_{83}x_3 & a_{84}x_4 & a_{85}x_5 & a_{86}x_6 & a_{87}x_7 & a_{88}x_8 \end{bmatrix}$$

Décomposition 1D par ligne

- n/p lignes de n éléments
- Si x et A sont stockés dans le processeur 0
  - Diffusion des n éléments de x aux (p-1) autres processeurs
  - Envoi de n/p lignes de n éléments de A à chacun des (p-1) autres processeurs
- Chaque processeur calcule n/p produits scalaires sur le vecteur entier
- Les composantes du vecteur y sont locales à chaque processeur
- Le processeur 0 reçoit n/p composantes de chacun des (p-1) autres processeurs

Polytech4 - Option Parallélisme - 2014 D. Etiemble 9

Produit matrice-vecteur – Décomposition 1D colonne

$$\begin{bmatrix} a_{11}x_1 & a_{12}x_2 & a_{12}x_3 & a_{12}x_4 & a_{12}x_5 & a_{12}x_6 & a_{12}x_7 & a_{12}x_8 \\ a_{21}x_1 & a_{22}x_2 & a_{23}x_3 & a_{24}y_4 & a_{25}x_5 & a_{26}x_6 & a_{27}x_7 & a_{28}x_8 \\ a_{31}x_1 & a_{32}x_2 & a_{33}x_3 & a_{34}x_4 & a_{35}x_5 & a_{36}y_6 & a_{37}x_7 & a_{38}x_8 \\ a_{41}x_1 & a_{42}x_2 & a_{43}x_3 & a_{44}x_4 & a_{45}x_5 & a_{46}x_6 & a_{47}x_7 & a_{48}x_8 \\ a_{51}x_1 & a_{52}x_2 & a_{53}x_3 & a_{54}x_4 & a_{55}x_5 & a_{56}x_6 & a_{57}x_7 & a_{58}x_8 \\ a_{61}x_1 & a_{62}x_2 & a_{63}x_3 & a_{64}x_4 & a_{65}x_5 & a_{66}x_6 & a_{67}x_7 & a_{68}x_8 \\ a_{71}x_1 & a_{72}x_2 & a_{73}x_3 & a_{74}x_4 & a_{75}x_5 & a_{76}x_6 & a_{77}x_7 & a_{78}x_8 \\ a_{81}x_1 & a_{82}x_2 & a_{83}x_3 & a_{84}x_4 & a_{85}x_5 & a_{86}x_6 & a_{87}x_7 & a_{88}x_8 \end{bmatrix}$$

Décomposition 1D par colonne

- n/p colonnes de n éléments
- Si x et A sont stockés dans le processeur 0
  - Envoi des n/p éléments de x aux (p-1) autres processeurs
  - Envoi de n/p colonnes de n éléments de A à chacun des (p-1) autres processeurs
- Chaque processeur calcule un vecteur colonne (calcul saxpy/daxpy sur n/p colonnes)
- Réduction des vecteurs colonne de chaque processeur dans le processeurs 0

Polytech4 - Option Parallélisme - 2014 D. Etiemble 10

Produit matrice-vecteur : décomposition 2 D

$$\begin{bmatrix}
a_{11}x_1 & a_{12}x_2 & a_{12}x_3 & a_{12}x_4 & a_{12}x_5 & a_{12}x_6 & a_{12}x_7 & a_{12}x_8 \\
a_{21}x_1 & a_{22}x_2 & a_{23}x_3 & a_{24}y_4 & a_{25}x_5 & a_{26}x_6 & a_{27}x_7 & a_{28}x_8 \\
a_{31}x_1 & a_{32}x_2 & a_{33}x_3 & a_{34}x_4 & a_{35}x_5 & a_{36}y_6 & a_{37}x_7 & a_{38}x_8 \\
a_{41}x_1 & a_{42}x_2 & a_{43}x_3 & a_{44}x_4 & a_{45}x_5 & a_{46}x_6 & a_{47}x_7 & a_{48}x_8 \\
a_{51}x_1 & a_{52}x_2 & a_{53}x_3 & a_{54}x_4 & a_{55}x_5 & a_{56}x_6 & a_{57}x_7 & a_{58}x_8 \\
a_{61}x_1 & a_{62}x_2 & a_{63}x_3 & a_{64}x_4 & a_{65}x_5 & a_{66}x_6 & a_{67}x_7 & a_{68}x_8 \\
a_{71}x_1 & a_{72}x_2 & a_{73}x_3 & a_{74}x_4 & a_{75}x_5 & a_{76}x_6 & a_{77}x_7 & a_{78}x_8 \\
a_{81}x_1 & a_{82}x_2 & a_{83}x_3 & a_{84}x_4 & a_{85}x_5 & a_{86}x_6 & a_{87}x_7 & a_{88}x_8
\end{bmatrix}$$

Décomposition 2D

- $n/p^{1/2}$ lignes/colonnes de $n/p^{1/2}$ éléments
- Si x et A sont stockés dans le processeur 0
  - $n/p^{1/2}$ diffusions de $n/p^{1/2}$ éléments de x le long de chaque colonne de $n/p^{1/2}$ processeurs
  - Envoi de n^2/p éléments de A à chacun des (p-1) autres processeurs
- Chaque processeur calcule $n/p^{1/2}$ composantes de vecteur colonne (soit par produit scalaire, soit par saxpy/daxpy)
- Réduction des vecteurs colonne de chaque ligne de processeurs et envoi au processeur 0

Multiplication matrice-matrice

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}$$

- Produit C=A.B ou A, B, C sont des matrices n x n
- Le produit de matrices peut être vu comme
  - n^2 produits scalaires
  - La somme de n produits externes
  - N produits matrice-vecteur
- On peut développer des algorithmes spécifiques

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \end{bmatrix}$$

Produits scalaires / Produits externes

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \end{bmatrix}$$

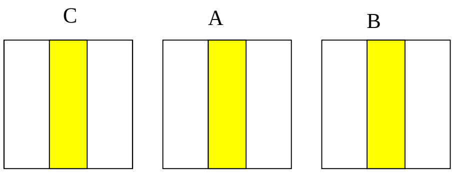
Produit matrice-vecteur

Produit matrice-matrice : algorithme 1D par colonnes

- Forme saxpy/daxpy par bloc
- Le processus k a les données A(:,k), B(:,k) et C(:,k)
- On exécute le produit de matrices dans l'ordre j, k, i
  - Pour j=1 à n

$$C(:,j) = C(:,j) + \sum_k A(:,k) \times B(k,j)$$

- Un processus a les données C et B dont il a besoin, mais doit accéder à toute la matrice A



- Algorithme
  - Chaque processus calcule la partie pour laquelle il a les données (avec m = myid)

$$C(:,m) = C(:,m) + A(:,m) * B(m,m)$$

  - Copier A(:,m) dans tampon S
  - Pour i=1 à p-1{
    - Envoyer S au processus (m+i) mod p
    - Recevoir S du processus (m-i) mod p

$$C(:,m) = C(:,m) + S * B[(m-i) \bmod p, m]$$

    - }

Produit matrices : algorithme 1D par colonnes (2)

- Exemple avec 9 processeurs

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \end{bmatrix}$$

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} \boxed{A_{00}B_{00}} + A_{01}B_{10} + A_{02}B_{20} & A_{00}B_{01} + \boxed{A_{01}B_{11}} + A_{02}B_{21} & A_{00}B_{02} + A_{01}B_{12} + \boxed{A_{02}B_{22}} \\ \boxed{A_{10}B_{00}} + A_{11}B_{10} + A_{12}B_{20} & A_{10}B_{01} + \boxed{A_{11}B_{11}} + A_{12}B_{21} & A_{10}B_{02} + A_{11}B_{12} + \boxed{A_{12}B_{22}} \\ \boxed{A_{20}B_{00}} + A_{21}B_{10} + A_{22}B_{20} & A_{20}B_{01} + \boxed{A_{21}B_{11}} + A_{22}B_{21} & A_{20}B_{02} + A_{21}B_{12} + \boxed{A_{22}B_{22}} \end{bmatrix}$$

Calcul local

$$\begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{00} & B_{11} & B_{22} \\ B_{00} & B_{11} & B_{22} \\ B_{00} & B_{11} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{00}B_{00} & A_{01}B_{11} & A_{02}B_{22} \\ A_{10}B_{00} & A_{11}B_{11} & A_{12}B_{22} \\ A_{20}B_{00} & A_{21}B_{11} & A_{22}B_{22} \end{bmatrix}$$

Produit de matrices : algorithme 1D par colonne (3)

i=1 P0 reçoit A(:,2), P1 reçoit A(:,0) et P2 reçoit A(:,1)

$$\begin{bmatrix} A_{01} & A_{02} & A_{00} \\ A_{11} & A_{12} & A_{10} \\ A_{21} & A_{22} & A_{20} \end{bmatrix} \times \begin{bmatrix} B_{10} & B_{21} & B_{02} \\ B_{10} & B_{21} & B_{02} \\ B_{10} & B_{21} & B_{02} \end{bmatrix} = \begin{bmatrix} A_{01}B_{10} & A_{02}B_{21} & A_{00}B_{02} \\ A_{11}B_{10} & A_{12}B_{21} & A_{10}B_{02} \\ A_{21}B_{10} & A_{22}B_{21} & A_{20}B_{02} \end{bmatrix}$$

i=2 P0 reçoit A(:,1), P1 reçoit A(:,2) et P2 reçoit A(:,0)

$$\begin{bmatrix} A_{02} & A_{00} & A_{01} \\ A_{12} & A_{10} & A_{11} \\ A_{22} & A_{20} & A_{21} \end{bmatrix} \times \begin{bmatrix} B_{20} & B_{01} & B_{12} \\ B_{20} & B_{01} & B_{12} \\ B_{20} & B_{01} & B_{12} \end{bmatrix} = \begin{bmatrix} A_{02}B_{20} & A_{00}B_{01} & A_{01}B_{12} \\ A_{12}B_{20} & A_{10}B_{01} & A_{11}B_{12} \\ A_{22}B_{20} & A_{20}B_{01} & A_{21}B_{12} \end{bmatrix}$$

i=1

i=2

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{00}B_{00} + A_{01}B_{10} + A_{02}B_{20} & A_{00}B_{01} + A_{01}B_{11} + A_{02}B_{21} & A_{00}B_{02} + A_{01}B_{12} + A_{02}B_{22} \\ A_{10}B_{00} + A_{11}B_{10} + A_{12}B_{20} & A_{10}B_{01} + A_{11}B_{11} + A_{12}B_{21} & A_{10}B_{02} + A_{11}B_{12} + A_{12}B_{22} \\ A_{20}B_{00} + A_{21}B_{10} + A_{22}B_{20} & A_{20}B_{01} + A_{21}B_{11} + A_{22}B_{21} & A_{20}B_{02} + A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

Local Local Local

Produit de matrices : algorithme de Cannon

- Algorithme 2D par blocs sur p = s^2 processeurs
  - Produit scalaire utilisé pour la boucle interne $C(i,j) = C(i,j) + \sum_k A(i,k) \times B(k,j)$
  - Boucle interne réécrite

$$C(i,j) = C(i,j) + \sum_k A\left[i,(i+j+k)\bmod(s)\right] \times B\left[(i+j+k)\bmod(s), j\right]$$

- Algorithme
  - Phase 1
    - Pour i=0 à s-1, décalage circulaire de la ligne i de la matrice A de i positions à gauche
    - Pour i=0 à s-1, décalage circulaire de la colonne j de la matrice B de j positions vers le haut
  - Phase 2
    - Pour k=0 à s-1, faire {
      - Pour i=0 à s-1 et pour j=0 à s-1, C(i,j)=C(i,j)+A(i,j)*B(i,j)
      - Décalage circulaire d'une position à gauche de chaque ligne de A
      - Décalage circulaire d'une position vers le haut de chaque colonne de B}
- Pour chaque bloc A(i,j)*B(i,j), produit de matrices local

Algorithme de Cannon (2)

- Exemple avec 9 processeurs. Les matrices C, A et B sont décomposées en blocs.

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \end{bmatrix}$$

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{00}B_{00} + A_{01}B_{10} + A_{02}B_{20} & A_{00}B_{01} + A_{01}B_{11} + A_{02}B_{21} & A_{00}B_{02} + A_{01}B_{12} + A_{02}B_{22} \\ A_{10}B_{00} + A_{11}B_{10} + A_{12}B_{20} & A_{10}B_{01} + A_{11}B_{11} + A_{12}B_{21} & A_{10}B_{02} + A_{11}B_{12} + A_{12}B_{22} \\ A_{20}B_{00} + A_{21}B_{10} + A_{22}B_{20} & A_{20}B_{01} + A_{21}B_{11} + A_{22}B_{21} & A_{20}B_{02} + A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

- Etape 1

$$\begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{11} & A_{12} & A_{10} \\ A_{22} & A_{20} & A_{21} \end{bmatrix}$$

Décalage circulaire
de la ligne i de i positions
vers la gauche

$$\begin{bmatrix} B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} B_{00} & B_{11} & B_{22} \\ B_{10} & B_{21} & B_{02} \\ B_{20} & B_{01} & B_{12} \end{bmatrix}$$

Décalage circulaire
de la colonne j de j positions
vers le haut

Polytech4 - Option
Parallélisme - 2014 D. Etiemble 17

Algorithme de Cannon (3)

Etape 2 : k=0

$$\begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{11} & A_{12} & A_{10} \\ A_{22} & A_{20} & A_{21} \end{bmatrix} \times \begin{bmatrix} B_{00} & B_{11} & B_{22} \\ B_{10} & B_{21} & B_{02} \\ B_{20} & B_{01} & B_{12} \end{bmatrix} = \begin{bmatrix} A_{00}B_{00} & A_{01}B_{11} & A_{02}B_{22} \\ A_{11}B_{10} & A_{12}B_{21} & A_{10}B_{02} \\ A_{22}B_{20} & A_{20}B_{01} & A_{21}B_{12} \end{bmatrix}$$

$$\begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{11} & A_{12} & A_{10} \\ A_{22} & A_{20} & A_{21} \end{bmatrix} \Rightarrow \begin{bmatrix} A_{01} & A_{02} & A_{00} \\ A_{12} & A_{10} & A_{11} \\ A_{20} & A_{21} & A_{22} \end{bmatrix}$$

Décalage circulaire
des lignes i de 1 position
vers la gauche

$$\begin{bmatrix} B_{00} & B_{11} & B_{22} \\ B_{10} & B_{21} & B_{02} \\ B_{20} & B_{01} & B_{12} \end{bmatrix} \Rightarrow \begin{bmatrix} B_{10} & B_{21} & B_{02} \\ B_{20} & B_{01} & B_{12} \\ B_{00} & B_{11} & B_{22} \end{bmatrix}$$

Décalage circulaire
des colonnes j de 1 position
vers le haut

Partie du calcul effectué

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} \boxed{A_{00}B_{00}} + A_{01}B_{10} + A_{02}B_{20} & A_{00}B_{01} + \boxed{A_{01}B_{11}} + A_{02}B_{21} & A_{00}B_{02} + A_{01}B_{12} + \boxed{A_{02}B_{22}} \\ A_{10}B_{00} + \boxed{A_{11}B_{10}} + A_{12}B_{20} & A_{10}B_{01} + A_{11}B_{11} + \boxed{A_{12}B_{21}} & \boxed{A_{10}B_{02}} + A_{11}B_{12} + A_{12}B_{22} \\ A_{20}B_{00} + A_{21}B_{10} + \boxed{A_{22}B_{20}} & \boxed{A_{20}B_{01}} + A_{21}B_{11} + A_{22}B_{21} & A_{20}B_{02} + \boxed{A_{21}B_{12}} + A_{22}B_{22} \end{bmatrix}$$

Polytech4 - Option
Parallélisme - 2014 D. Etiemble 18

Algorithme de Cannon (4)

Etape 2 : k=1

$$\begin{bmatrix} A_{01} & A_{02} & A_{00} \\ A_{12} & A_{10} & A_{11} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{10} & B_{21} & B_{02} \\ B_{20} & B_{01} & B_{12} \\ B_{00} & B_{11} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{01}B_{10} & A_{02}B_{21} & A_{00}B_{02} \\ A_{12}B_{20} & A_{10}B_{01} & A_{11}B_{12} \\ A_{20}B_{00} & A_{21}B_{11} & A_{22}B_{22} \end{bmatrix}$$

$$\begin{bmatrix} A_{01} & A_{02} & A_{00} \\ A_{12} & A_{10} & A_{11} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} A_{02} & A_{00} & A_{01} \\ A_{10} & A_{11} & A_{12} \\ A_{21} & A_{22} & A_{20} \end{bmatrix}$$

Décalage circulaire
des lignes i de 1 position
vers la gauche

$$\begin{bmatrix} B_{10} & B_{21} & B_{02} \\ B_{20} & B_{01} & B_{12} \\ B_{00} & B_{11} & B_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} B_{20} & B_{01} & B_{12} \\ B_{00} & B_{11} & B_{22} \\ B_{10} & B_{21} & B_{02} \end{bmatrix}$$

Décalage circulaire
des colonnes j de 1 position
vers le haut

Partie du calcul effectué

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{00}B_{00} + \boxed{A_{01}B_{10}} + A_{02}B_{20} & A_{00}B_{01} + A_{01}B_{11} + \boxed{A_{02}B_{21}} & \boxed{A_{00}B_{02}} + A_{01}B_{12} + A_{02}B_{22} \\ A_{10}B_{00} + A_{11}B_{10} + \boxed{A_{12}B_{20}} & \boxed{A_{10}B_{01}} + A_{11}B_{11} + A_{12}B_{21} & A_{10}B_{02} + \boxed{A_{11}B_{12}} + A_{12}B_{22} \\ \boxed{A_{20}B_{00}} + A_{21}B_{10} + A_{22}B_{20} & A_{20}B_{01} + \boxed{A_{21}B_{11}} + A_{22}B_{21} & A_{20}B_{02} + A_{21}B_{12} + \boxed{A_{22}B_{22}} \end{bmatrix}$$

Algorithme de Cannon (5)

Etape 2 : k=2

$$\begin{bmatrix} A_{02} & A_{00} & A_{01} \\ A_{10} & A_{11} & A_{12} \\ A_{21} & A_{22} & A_{20} \end{bmatrix} \times \begin{bmatrix} B_{20} & B_{01} & B_{12} \\ B_{00} & B_{11} & B_{22} \\ B_{10} & B_{21} & B_{02} \end{bmatrix} = \begin{bmatrix} A_{02}B_{20} & A_{00}B_{01} & A_{01}B_{12} \\ A_{10}B_{00} & A_{11}B_{11} & A_{12}B_{22} \\ A_{21}B_{10} & A_{22}B_{21} & A_{20}B_{02} \end{bmatrix}$$

$$\begin{bmatrix} A_{02} & A_{00} & A_{01} \\ A_{10} & A_{11} & A_{12} \\ A_{21} & A_{22} & A_{20} \end{bmatrix} \Rightarrow \begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix}$$

Décalage circulaire
des lignes i de 1 position
vers la gauche

Matrice A originale

$$\begin{bmatrix} B_{20} & B_{01} & B_{12} \\ B_{00} & B_{11} & B_{22} \\ B_{10} & B_{21} & B_{02} \end{bmatrix} \Rightarrow \begin{bmatrix} B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \end{bmatrix}$$

Décalage circulaire
des colonnes j de 1 position
vers le haut

Matrice B originale

Partie du calcul effectué

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{00}B_{00} + A_{01}B_{10} + \boxed{A_{02}B_{20}} & \boxed{A_{00}B_{01}} + A_{01}B_{11} + A_{02}B_{21} & A_{00}B_{02} + \boxed{A_{01}B_{12}} + A_{02}B_{22} \\ \boxed{A_{10}B_{00}} + A_{11}B_{10} + A_{12}B_{20} & A_{10}B_{01} + \boxed{A_{11}B_{11}} + A_{12}B_{21} & A_{10}B_{02} + A_{11}B_{12} + \boxed{A_{12}B_{22}} \\ A_{20}B_{00} + \boxed{A_{21}B_{10}} + A_{22}B_{20} & A_{20}B_{01} + A_{21}B_{11} + \boxed{A_{22}B_{21}} & \boxed{A_{20}B_{02}} + A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

Produits de matrices : algorithme de Fox et al

- Algorithme 2D par blocs sur p = s^2 processeurs
  - Produit scalaire utilisé pour la boucle interne

$$C(i,j) = C(i,j) + \sum_k A(i,k) \times B(k,j)$$

- Algorithme
  - Pour k=0 à s-1, faire {
    - Le processus dans la ligne i contenant A[i, (i+k) mod s] diffuse A[i, (i+k) mod s] à tous les autres processus de la même ligne i
    - Les processus dans la ligne i reçoivent A[i, (i+k) mod s] dans une matrice locale T
    - Pour i=0 à s-1 et pour j=0 à s-1, C(i,j)=C(i,j)+T(i,j)*B(i,j)
    - Décalage circulaire d'une position vers le haut de chaque colonne de B}
- Pour chaque bloc A(i,j)*B(i,j), produit de matrices local

Algorithme de Fox (2)

- Exemple avec 9 processeurs. Les matrices C, A et B sont décomposées en blocs.

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \end{bmatrix}$$

- k=0

$$\begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \Rightarrow \overset{T}{\begin{bmatrix} A_{00} & A_{00} & A_{00} \\ A_{11} & A_{11} & A_{11} \\ A_{22} & A_{22} & A_{22} \end{bmatrix}}$$

$$\begin{bmatrix} A_{00} & A_{00} & A_{00} \\ A_{11} & A_{11} & A_{11} \\ A_{22} & A_{22} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{00}B_{00} & A_{00}B_{01} & A_{00}B_{02} \\ A_{11}B_{10} & A_{11}B_{11} & A_{11}B_{12} \\ A_{22}B_{20} & A_{22}B_{21} & A_{22}B_{22} \end{bmatrix}$$

$$\begin{bmatrix} B_{00} & B_{01} & B_{22} \\ B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \\ B_{00} & B_{01} & B_{02} \end{bmatrix}$$

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} \boxed{A_{00}B_{00}} + A_{01}B_{10} + A_{02}B_{20} & \boxed{A_{00}B_{01}} + A_{01}B_{11} + A_{02}B_{21} & \boxed{A_{00}B_{02}} + A_{01}B_{12} + A_{02}B_{22} \\ A_{10}B_{00} + \boxed{A_{11}B_{10}} + A_{12}B_{20} & A_{10}B_{01} + \boxed{A_{11}B_{11}} + A_{12}B_{21} & A_{10}B_{02} + \boxed{A_{11}B_{12}} + A_{12}B_{22} \\ A_{20}B_{00} + A_{21}B_{10} + \boxed{A_{22}B_{20}} & A_{20}B_{01} + A_{21}B_{11} + \boxed{A_{22}B_{21}} & A_{20}B_{02} + A_{21}B_{12} + \boxed{A_{22}B_{22}} \end{bmatrix}$$

Algorithme de Fox (3)

- k=1

$$\begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} A_{01} & A_{01} & A_{01} \\ A_{12} & A_{12} & A_{12} \\ A_{20} & A_{20} & A_{20} \end{bmatrix}$$

$$\begin{bmatrix} A_{01} & A_{01} & A_{01} \\ A_{12} & A_{12} & A_{12} \\ A_{20} & A_{20} & A_{20} \end{bmatrix} \times \begin{bmatrix} B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \\ B_{00} & B_{01} & B_{02} \end{bmatrix} = \begin{bmatrix} A_{01}B_{10} & A_{01}B_{11} & A_{01}B_{12} \\ A_{12}B_{20} & A_{12}B_{21} & A_{12}B_{22} \\ A_{20}B_{00} & A_{20}B_{01} & A_{20}B_{02} \end{bmatrix}$$

$$\begin{bmatrix} B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \\ B_{00} & B_{01} & B_{02} \end{bmatrix} \Rightarrow \begin{bmatrix} B_{20} & B_{21} & B_{22} \\ B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \end{bmatrix}$$

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{00}B_{00} + \boxed{A_{01}B_{10}} + A_{02}B_{20} & A_{00}B_{01} + \boxed{A_{01}B_{11}} + A_{02}B_{21} & A_{00}B_{02} + \boxed{A_{01}B_{12}} + A_{02}B_{22} \\ A_{10}B_{00} + A_{11}B_{10} + \boxed{A_{12}B_{20}} & A_{10}B_{01} + A_{11}B_{11} + \boxed{A_{12}B_{21}} & A_{10}B_{02} + A_{11}B_{12} + \boxed{A_{12}B_{22}} \\ \boxed{A_{20}B_{00}} + A_{21}B_{10} + A_{22}B_{20} & \boxed{A_{20}B_{01}} + A_{21}B_{11} + A_{22}B_{21} & \boxed{A_{20}B_{02}} + A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

Algorithme de Fox (4)

- k=2

$$\begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} A_{02} & A_{02} & A_{02} \\ A_{10} & A_{10} & A_{10} \\ A_{21} & A_{21} & A_{21} \end{bmatrix}$$

$$\begin{bmatrix} A_{02} & A_{02} & A_{02} \\ A_{10} & A_{10} & A_{10} \\ A_{21} & A_{21} & A_{21} \end{bmatrix} \times \begin{bmatrix} B_{20} & B_{21} & B_{22} \\ B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \end{bmatrix} = \begin{bmatrix} A_{02}B_{20} & A_{02}B_{21} & A_{00}B_{22} \\ A_{10}B_{00} & A_{10}B_{01} & A_{10}B_{02} \\ A_{21}B_{10} & A_{21}B_{11} & A_{21}B_{12} \end{bmatrix}$$

$$\begin{bmatrix} B_{20} & B_{21} & B_{22} \\ B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \end{bmatrix} \Rightarrow \begin{bmatrix} B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \\ B_{00} & B_{01} & B_{02} \end{bmatrix}$$

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{00}B_{00} + A_{01}B_{10} + \boxed{A_{02}B_{20}} & A_{00}B_{01} + A_{01}B_{11} + \boxed{A_{02}B_{21}} & A_{00}B_{02} + A_{01}B_{12} + \boxed{A_{02}B_{22}} \\ \boxed{A_{10}B_{00}} + A_{11}B_{10} + A_{12}B_{20} & \boxed{A_{10}B_{01}} + A_{11}B_{11} + A_{12}B_{21} & \boxed{A_{10}B_{02}} + A_{11}B_{12} + A_{12}B_{22} \\ A_{20}B_{00} + \boxed{A_{21}B_{10}} + A_{22}B_{20} & A_{20}B_{01} + \boxed{A_{21}B_{11}} + A_{22}B_{21} & A_{20}B_{02} + \boxed{A_{21}B_{12}} + A_{22}B_{22} \end{bmatrix}$$

** here Solving systems of linear equations: *Ax=b*

Dense matrices

Direct Methods:

Gaussian Elimination – seq. time complexity O(n^3)
LU-Decomposition – seq. time complexity O(n^3)

Sparse matrices (with good convergence properties)

Iterative Methods:

Jacobi iteration
Gauss-Seidel relaxation (not good for parallelization)
Red-Black ordering
Multigrid

Gauss Elimination

- Solve $Ax = b$
- Consists of two phases:
  - –**Forward elimination**
  - –**Back substitution**
- *Forward Elimination* reduces $Ax = b$ to an upper triangular system $Tx = b'$
- *Back substitution* can then solve $Tx = b'$ for x

$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array}\right]$$

$$\Downarrow$$

$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ 0 & a'_{22} & a'_{23} & b'_2 \\ 0 & 0 & a''_{33} & b''_3 \end{array}\right]$$

$$\Downarrow$$

$$x_3 = \frac{b''_3}{a''_{33}} \qquad x_2 = \frac{b'_2 - a'_{23}x_3}{a'_{22}}$$

$$x_1 = \frac{b_1 - a_{13}x_3 - a_{12}x_2}{a_{11}}$$

Forward Elimination

Back Substitution

Forward Elimination

	$x_1 - x_2 + x_3 = 6$
$-(3/1)$	$3x_1 + 4x_2 + 2x_3 = 9$
$-(2/1)$	$2x_1 + x_2 + x_3 = 7$

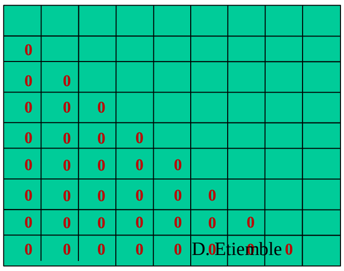
$\Rightarrow$

	$x_1 - x_2 + x_3 = 6$
	$0 \; +7x_2 - x_3 = -9$
$-(3/7)$	$0 \; + 3x_2 - x_3 = -5$

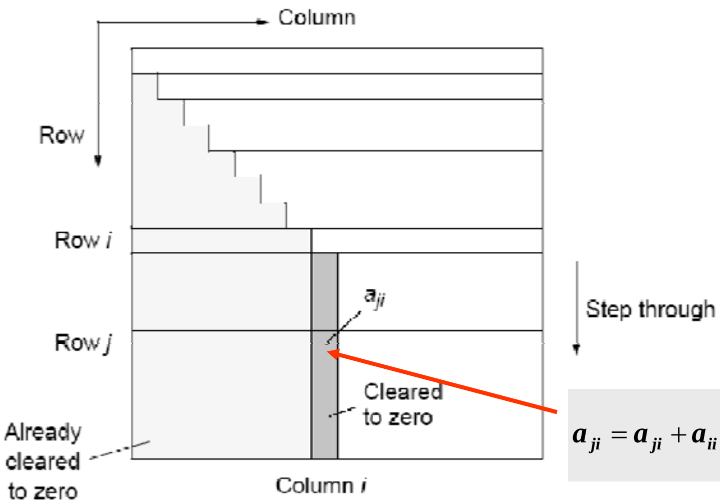
$\Rightarrow$

$x_1 - x_2 + x_3 = 6$
$0 \quad 7x_2 - x_3 = -9$
$0 \quad 0 \; -(4/7)x_3 = -(8/7)$

Solve using BACK SUBSTITUTION: $x_3 = 2$ $x_2 = -1$ $x_1 = 3$



Forward Elimination



$$a_{ji} = a_{ji} + a_{ii}\left(\frac{-a_{ji}}{a_{ii}}\right) = 0$$

Gaussian Elimination

MULTIPLIERS

	$4x_0$	$+6x_1$	$+2x_2$	$-2x_3$	$=$	8
-(2/4)	$2x_0$		$+5x_2$	$-2x_3$	$=$	4
-(-4/4)	$-4x_0$	$-3x_1$	$-5x_2$	$+4x_3$	$=$	1
-(8/4)	$8x_0$	$+18x_1$	$-2x_2$	$+3x_3$	$=$	40

Gaussian Elimination

MULTIPLIERS

	$4x_0$	$+6x_1$	$+2x_2$	$-2x_3$	$=$	8
		$-3x_1$	$+4x_2$	$-1x_3$	$=$	0
-(3/-3)		$+3x_1$	$-3x_2$	$+2x_3$	$=$	9
-(6/-3)		$+6x_1$	$-6x_2$	$+7x_3$	$=$	24

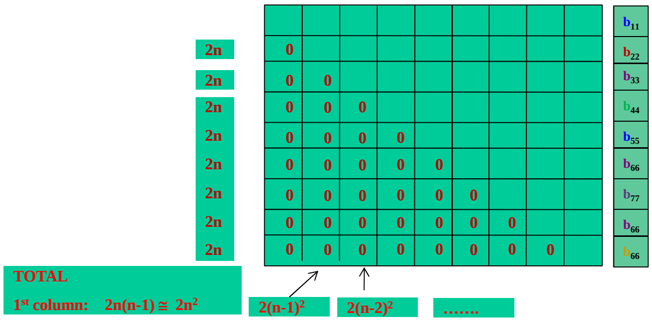
Gaussian Elimination

M	$4x_0$	$+6x_1$	$+2x_2$	$-2x_3$	$=$	8
U						
L		$-3x_1$	$+4x_2$	$-1x_3$	$=$	0
T						
I			$1x_2$	$+1x_3$	$=$	9
P						
L						
I						
E	??		$2x_2$	$+5x_3$	$=$	24
R						

Gaussian Elimination

$4x_0$	$+6x_1$	$+2x_2$	$-2x_3$	$=$	8
	$-3x_1$	$+4x_2$	$-1x_3$	$=$	0
		$1x_2$	$+1x_3$	$=$	9
			$3x_3$	$=$	6

Operation count in Forward Elimination



TOTAL # of Operations for FORWARD ELIMINATION :

$$2n^2 + 2(n-1)^2 + \ldots + 2*(2)^2 + 2*(1)^2 = 2\sum_{i=1}^{n} i^2$$

$$= 2\frac{n(n+1)(2n+1)}{6}$$

$$= O(n^3)$$

Back Substitution
(* Pseudocode *)

for $i \leftarrow n - 1$ down to 1 do

/* calculate x_i */
$x[i] \leftarrow b[i] / a[i, i]$

/* substitute in the equations above */
for $j \leftarrow 0$ to $i - 1$ **do**
$b[j] \leftarrow b[j] - x[i] \times a[j, i]$
endfor
endfor

Time Complexity? ➔➔ *$O(n^2)$*

Partial Pivoting

If $a_{i,i}$ is zero or close to zero, we will not be able to compute the quantity $-a_{j,i} / a_{i,i}$

Procedure must be modified into so-called ***partial pivoting*** by **swap**ping the i^{th} row with the row below it that has the **largest** absolute element in the i^{th} column of any of the rows below the i^{th} row (if there is one).

(Reordering equations will not affect the system.)

Sequential Code

Without partial pivoting:

```
for (i = 0; i < n-1; i++)            /* for each row, except last */
    for (j = i+1; j < n; j++) {        /* step thro subsequent rows */

        m = a[j][i]/a[i][i];         /* Compute multiplier  */

        for (k = i; k < n; k++)      /* last n-i-1 elements of row j */
          a[j][k] = a[j][k] - a[i][k] * m;

        b[j] = b[j] - b[i] * m;        /* modify right side */
    }
```

The time complexity: $T_{seq} = O(n^3)$

Computing the Determinant

Given an upper triangular system of equations

$$D = t_{00}\, t_{11} \ldots t_{n-1,n-1}$$

$$D = \begin{vmatrix} t_{00} & t_{01} & \cdots & t_{0,n-1} \\ 0 & t_{11} & \cdots & t_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & t_{n-1,n-1} \end{vmatrix}$$

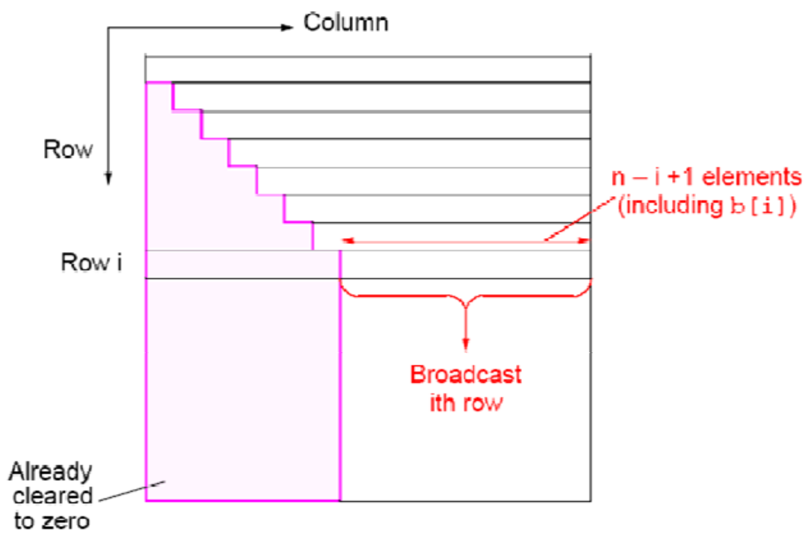
If ***pivoting*** is used then

$D = t_{00}\, t_{11} \ldots t_{n-1,n-1} (-1)^p$ *where p is the number of times the rows are pivoted*

Singular systems

- When two equations are identical, we would loose one degree of freedom
  n-1 equations for *n* unknowns ➔ infinitely many solutions
- This is difficult to find out for large sets of equations.
  The fact that the **determinant** of a singular system is **zero** can be used and tested after the elimination stage.

Parallel implementation



Time Complexity Analysis (P = n)

Communication

(n-1) broadcasts performed sequentially - i^{th} broadcast contains (n-i) elements.
Total Time: $T_{par} = O(n^2)$ (How ?)

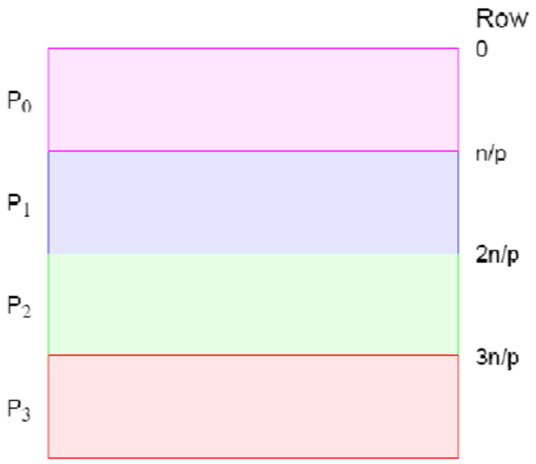
Computation

After row i is broadcast, each processor P_j will compute its multiplier, and operate upon n-j+2 elements of its row. Ignoring the computation of the multiplier, there are (n-j+2) multiplications and (n-j+2) subtractions.
Total Time: $T_{par} = O(n^2)$

Therefore, $$T_{par} = O(n^2)$$

Efficiency will be relatively low because all the processors before the processor holding row i do not participate in the computation again.

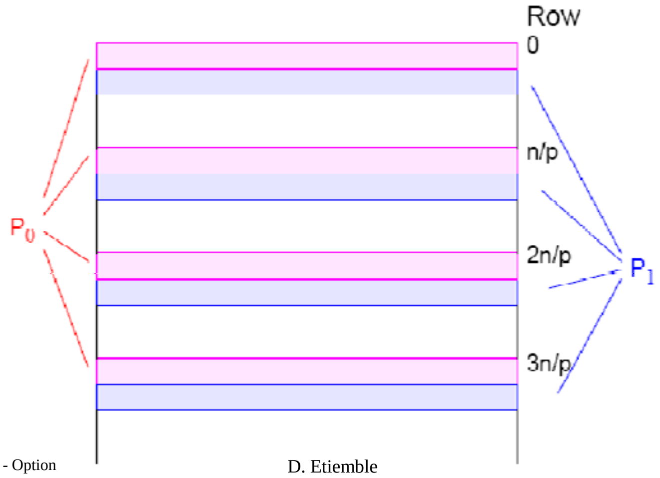
Strip Partitioning



Poor processor allocation!
Processors do not participate in computation after their last row is processed.

Cyclic-Striped Partitioning

An alternative which equalizes the processor workload



Jacobi Iterative Method (Sequential)

Iterative methods provide an **alternative** to the *elimination methods*.

$$\mathbf{Ax = b} \qquad \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix}$$

$$[\mathbf{D} + (\mathbf{A} - \mathbf{D})]\mathbf{x} = \mathbf{b} \quad \Rightarrow \quad \mathbf{Dx} = \mathbf{b} - (\mathbf{A} - \mathbf{D})\mathbf{x} \quad \Rightarrow \quad \mathbf{x} = \mathbf{D}^{-1}[\mathbf{b} - (\mathbf{A} - \mathbf{D})\mathbf{x}]$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_{new} = \begin{bmatrix} 1/a_{11} & 0 & 0 \\ 0 & 1/a_{22} & 0 \\ 0 & 0 & 1/a_{33} \end{bmatrix} * \left(\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} - \begin{bmatrix} 0 & a_{12} & a_{13} \\ a_{21} & 0 & a_{23} \\ a_{31} & a_{32} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_{old} \right)$$

$$x_1^k = \frac{b_1 - a_{12}x_2^{k-1} - a_{13}x_3^{k-1}}{a_{11}} \qquad x_2^k = \frac{b_2 - a_{21}x_1^{k-1} - a_{23}x_3^{k-1}}{a_{22}} \qquad x_3^k = \frac{b_3 - a_{31}x_1^{k-1} - a_{32}x_2^{k-1}}{a_{33}}$$

Choose an initial guess (i.e. all zeros) and Iterate until the equality is satisfied.
No guarantee for convergence! Each iteration takes $O(n^2)$ time!

Gauss-Seidel Method (Sequential)

- The *Gauss-Seidel* method is a commonly used *iterative method*.
- It is same as **Jacobi technique** except with one important difference:
  A newly computed *x* value (say x_k) is substituted in the subsequent equations (equations *k+1, k+2, …, n*) **in the same iteration**.

Example: Consider the *3x3* system below:

$$x_1^{new} = \frac{b_1 - a_{12}x_2^{old} - a_{13}x_3^{old}}{a_{11}}$$

$$x_2^{new} = \frac{b_2 - a_{21}x_1^{new} - a_{23}x_3^{old}}{a_{22}}$$

$$x_3^{new} = \frac{b_3 - a_{31}x_1^{new} - a_{32}x_2^{new}}{a_{33}}$$

$$\{X\}_{old} \leftarrow \{X\}_{new}$$

- First, choose initial guesses for the x's.
- A simple way to obtain initial guesses is to assume that they are all **zero**.
- Compute **new** $\mathbf{x_1}$ using the previous iteration values.
- **New** x_1 is substituted in the equations to calculate x_2 and x_3
- The process is repeated for $\mathbf{x_2}$, $\mathbf{x_3}$, …

Convergence Criterion for Gauss-Seidel Method

- Iterations are repeated until the convergence criterion is satisfied:

$$|\varepsilon_{a,i}| = \left|\frac{x_i^j - x_i^{j-1}}{x_i^j}\right| 100\% \prec \varepsilon_s$$

For all *i*, where *j* and *j-1* are the *current* and *previous* iterations.

- As any other iterative method, the **Gauss-Seidel** method has problems:
  - It may not converge or it converges very slowly.
- If the coefficient matrix A is **Diagonally Dominant** Gauss-Seidel is guaranteed to converge.

Diagonally Dominant ➔

For each equation i :

$$|a_{ii}| \succ \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{i,j}|$$

- Note that this is not a necessary condition, i.e. the system *may* still have a chance to converge even if A is not diagonally dominant.

Time Complexity: Each iteration takes O(n^2)