

**Parallélisme**

COURS 7 - ALGORITHMIQUE SUR ANNEAU DE PROCESSEURS

ERIC GOUBAULT  
COMMISSARIAT À L'ÉNERGIE ATOMIQUE & CHAIRE ÉCOLE  
POLYTECHNIQUE/THALÈS  
SACLAY

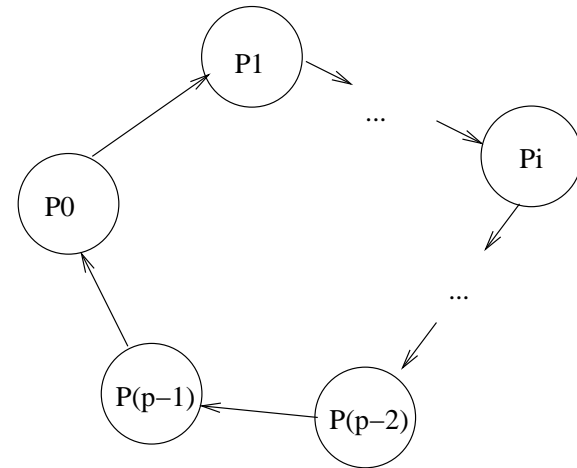
LE 3 MARS 2010

1

ALGORITHMIQUE SUR ANNEAU DE PROCESSEURS

- Macro-communications sur un anneau
- Produit matrice-vecteur
- Factorisation LU

ARCHITECTURE



3

ARCHITECTURE

- $p$  processeurs en anneau
- chacun a accès à:
  - son numéro d'ordre (entre 0 et  $p - 1$ ), par `my_num()`
  - nombre total de processeur: `tot_proc_num (=p)`

## FONCTIONNEMENT

Mode SPMD:

- tous les processeurs exécutent le même code,
- ils calculent tous dans leur mémoire locale,
- ils peuvent envoyer un message au processeur de numéro `my_num()+1[p]` par `send(adr,L)` avec,
  - `adr`, adresse de la première valeur dans la mémoire locale de l'expéditeur
  - `L` la longueur du message
- ils peuvent recevoir un message de `my_num()-1[p]` par `receive(adr,L)`

On doit s'arranger pour qu'à tout `send` corresponde un `receive`.

5

## SÉMANTIQUE

Plusieurs hypothèses possibles:

- `send` et `receive` bloquants (OCCAM etc.)
- plus classiquement `send` non bloquant mais `receive` bloquant (mode par défaut en PVM, MPI)
- plus moderne: aucun bloquant (trois threads en fait: 1 pour calcul, 1 pour `send`, 1 pour `receive`)

## MODÉLISATION DU COÛT D'UNE COMMUNICATION

Difficile en général: ici envoyer/recevoir un message de longueur  $L$  (au voisin immédiat) coûtera:

$$\beta + L\tau$$

où:

- $\beta$  est le coût d'initialisation (latence)
- $\tau$  (débit) mesure la vitesse de transmission en régime permanent

D'où envoyer/recevoir un message de longueur  $L$  de `my_num()+/-q` coûte  $q(\beta + L\tau)$ .

7

## PROBLÈME ÉLÉMENTAIRE: LA DIFFUSION

- C'est l'envoi par un  $P_k$  d'un message de longueur  $L$  (stocké à l'adresse `adr`) à tous les autres processeurs:
- Implémenté de façon efficace dans la plupart des bibliothèques de communications (PVM, MPI etc.).

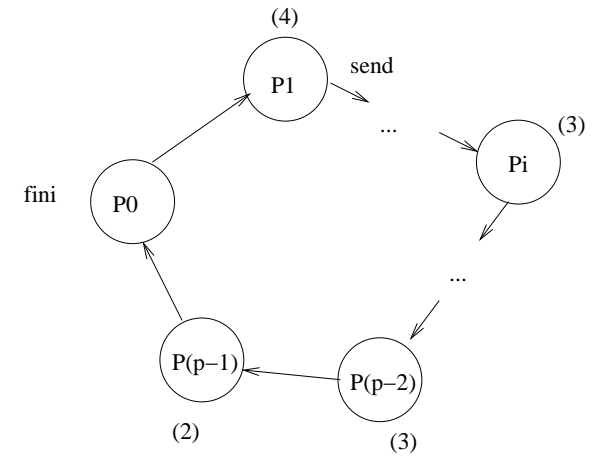
IMPLÉMENTATION (RECEIVE bloquant)

```

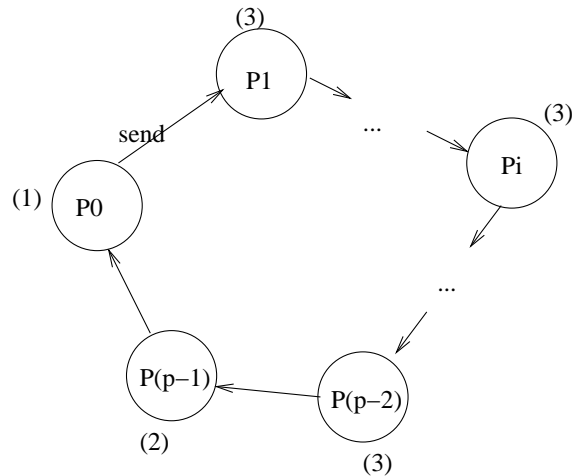
broadcast(k,adr,L) { // emetteur initial=k
  q = my_num();
  p = tot_proc_num();
  if (q == k)
    (1) send(adr,L);
  else
    if (q == k-1 mod p)
      (2) receive(adr,L);
    else {
      (3) receive(adr,L);
      (4) send(adr,L);
    }
}

```

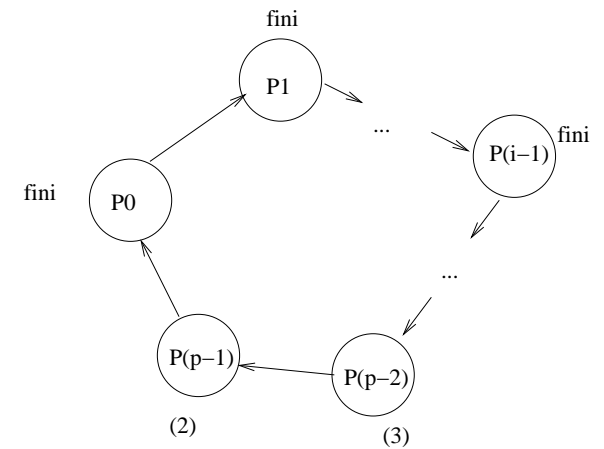
EXÉCUTION - TEMPS  $\beta + L\tau$



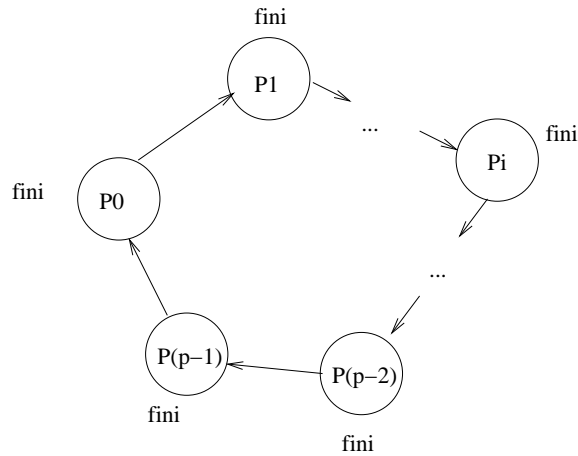
EXÉCUTION - TEMPS 0 ET  $\kappa=0$



EXÉCUTION - TEMPS  $i(\beta + L\tau)$  ( $i < p - 1$ )



EXÉCUTION - TEMPS  $(p - 1)(\beta + L\tau)$



13

PROGRAMME

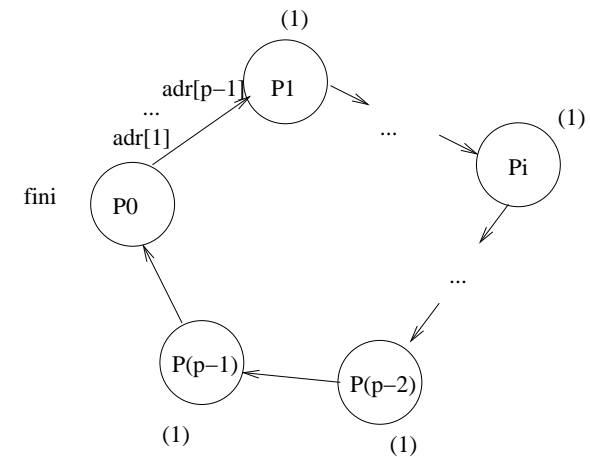
```
scatter(k,adr,L) {
  q = my_num();
  p = tot_proc_num();
  if (q == k) {
    adr = adr[k];
    for (i=1;i<p;i=i+1)
      send(adr[k-i mod p],L); }
  else
    (1) receive(adr,L);
  for (i=1;i<k-q mod p;i = i+1) {
    (2) send(adr,L);
    (3) receive(temp,L);
    adr = temp; } }
```

15

DIFFUSION PERSONNALISÉE

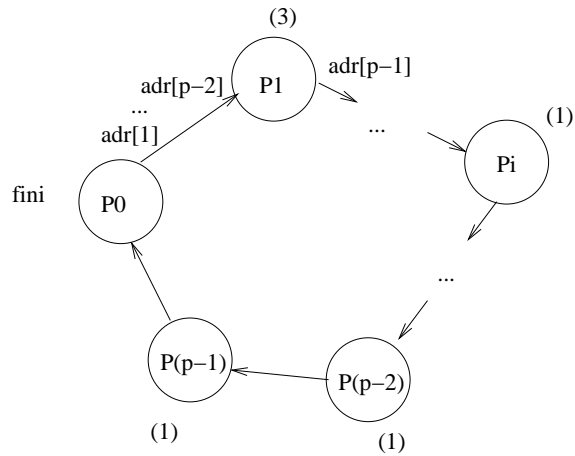
- **send** non-bloquant, **receive** bloquant
- envoi par  $P_k$  d'un message différent à tous les processeurs (en  $adr[q]$  dans  $P_k$  pour  $P_q$ )
- à la fin chaque processeur a son message à la location **adr**
- opère en pipeline: recouvrement entre les différentes communications!

EXÉCUTION - TEMPS 0 ET  $\kappa=0$

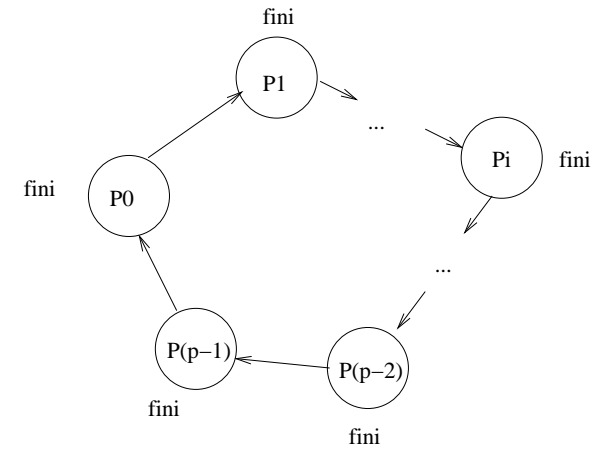


16

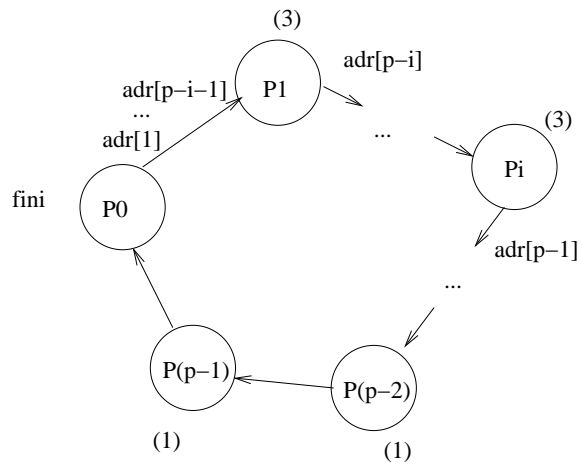
EXÉCUTION - TEMPS  $\beta + L\tau$



EXÉCUTION - TEMPS  $(p - 1)(\beta + L\tau)$



EXÉCUTION - TEMPS  $i(\beta + L\tau)$



**ECHANGE TOTAL**

- Chaque processeur  $k$  veut envoyer un message à tous les autres
- Au départ chaque processeur dispose de son message à envoyer à tous les autres à la location **my\_adr**
- A la fin, tous ont un tableau (le même) **adr[]** tel que **adr[q]** contient le message envoyé par le processeur **q**

Peut se faire aussi en  $(p - 1)(\beta + L\tau)$ . De même pour l'échange total personnalisé

## PROGRAMME

```
all-to-all(my_adr,adr,L) {
  q = my_num();
  p = tot_proc_num();
  adr[q] == my_adr;
  for (i=1;i<p;i++) {
    send(adr[q-i+1 mod p],L);
    receive(adr[q-i mod p],L);
  }
}
```

21

## DIFFUSION PIPELINÉE

Les temps d'une diffusion simple et d'une diffusion personnalisée sont les mêmes; peut-on améliorer le temps de la diffusion simple en utilisant les idées de la diffusion personnalisée?

- tronçonner le message à envoyer en  $r$  morceaux ( $r$  divise  $L$ )
- l'émetteur envoie successivement les  $r$  morceaux, avec recouvrement partiel des communications
- au début ces morceaux de messages sont dans  $\text{adr}[1], \dots, \text{adr}[r]$  du processeur  $k$

## PROGRAMME

```
broadcast(k,adr,L) {
  q = my_num();
  p = tot_proc_num();
  if (q == k)
    for (i=1;i<=r;i++) send(adr[i],L/r);
  else
    if (q == k-1 mod p)
      for (i=1;i<=r;i++) receive(adr[i],L/r);
    else {
      receive(adr[1],L/r);
      for (i=1;i<r;i++) {
        send(adr[i],L/r);
        receive(adr[i+1],L/r); } } }
```

23

## TEMPS D'EXÉCUTION

- le premier morceau de longueur  $L/r$  du message sera arrivé au dernier processeur  $k-1 \bmod p$  en temps  $(p-1)(\beta + \frac{L}{r}\tau)$  (diffusion simple)
- les  $r-1$  autres morceaux arrivent les uns derrière les autres, d'où un temps supplémentaire de  $(r-1)(\beta + \frac{L}{r}\tau)$
- En tout  $(p-2+r)(\beta + \frac{L}{r}\tau)$

## OPTIMISATION DU PARAMÈTRE $r$

- $r_{opt} = \sqrt{\frac{L(p-2)\tau}{\beta}}$
- le temps optimal d'exécution est donc de
$$\left(\sqrt{(p-2)\beta} + \sqrt{L\tau}\right)^2$$
- quand  $L$  tend vers l'infini, ceci est asymptotiquement équivalent à  $L\tau$ , le facteur  $p$  devient négligeable!

25

## PRODUIT MATRICE-VECTEUR

Problème: calculer  $y = Ax$  avec,

- $A$  matrice de dimension  $n \times n$
- $x$  vecteur à  $n$  composantes (de 0 à  $n - 1$ )
- sur un anneau de  $p$  processeurs, avec  $r = n/p$  entier

## PROGRAMME SÉQUENTIEL

le calcul produit matrice-vecteur revient au calcul de  $n$  produits scalaires:

```
for (i=1;i<=n;i++)
  for (j=1;j<=n;j++)
    y[i] = y[i]+a[i,j]*x[j];
```

27

## PRINCIPE DE LA DISTRIBUTION

Distribuer le calcul des produits scalaires aux processeurs:

- chaque processeur a en mémoire  $r$  lignes de la matrice  $\mathbf{A}$  rangées dans une matrice  $\mathbf{a}$  de dimension  $r \times n$
- $P_q$  contient les lignes  $qr$  à  $(q+1)r - 1$  de la matrice  $\mathbf{A}$  et les composantes de même rang des vecteurs  $\mathbf{x}$  et  $\mathbf{y}$ :

```
float a[r][n];
float x[r],y[r];
```

PRINCIPE DU CALCUL -DISTRIBUTION INITIALE DES DONNÉES

$$P_0 \left( \begin{array}{cccccccc} A_{00} & A_{01} & A_{02} & A_{03} & A_{04} & A_{05} & A_{06} & A_{07} \\ A_{10} & A_{11} & A_{12} & A_{13} & A_{14} & A_{15} & A_{16} & A_{17} \end{array} \right) \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$

$$P_1 \left( \begin{array}{cccccccc} A_{20} & A_{21} & A_{22} & A_{23} & A_{24} & A_{25} & A_{26} & A_{27} \\ A_{30} & A_{31} & A_{32} & A_{33} & A_{34} & A_{35} & A_{36} & A_{37} \end{array} \right) \begin{pmatrix} x_2 \\ x_3 \end{pmatrix}$$

$$P_2 \left( \begin{array}{cccccccc} A_{40} & A_{41} & A_{42} & A_{43} & A_{44} & A_{45} & A_{46} & A_{47} \\ A_{50} & A_{51} & A_{52} & A_{53} & A_{54} & A_{55} & A_{56} & A_{57} \end{array} \right) \begin{pmatrix} x_4 \\ x_5 \end{pmatrix}$$

$$P_3 \left( \begin{array}{cccccccc} A_{60} & A_{61} & A_{62} & A_{63} & A_{64} & A_{65} & A_{66} & A_{67} \\ A_{70} & A_{71} & A_{72} & A_{73} & A_{74} & A_{75} & A_{76} & A_{77} \end{array} \right) \begin{pmatrix} x_6 \\ x_7 \end{pmatrix}$$

29

DEUXIÈME ÉTAPE

$$P_0 \left( \begin{array}{cccccccc} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{06} & A_{07} \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{16} & A_{17} \end{array} \right) \begin{pmatrix} x_6 \\ x_7 \end{pmatrix} \text{ temp} \leftarrow \begin{pmatrix} x_4 \\ x_5 \end{pmatrix}$$

$$P_1 \left( \begin{array}{cccccccc} A_{20} & A_{21} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ A_{30} & A_{31} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{array} \right) \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \text{ temp} \leftarrow \begin{pmatrix} x_6 \\ x_7 \end{pmatrix}$$

$$P_2 \left( \begin{array}{cccccccc} \bullet & \bullet & A_{42} & A_{43} & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & A_{52} & A_{53} & \bullet & \bullet & \bullet & \bullet \end{array} \right) \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} \text{ temp} \leftarrow \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$

$$P_3 \left( \begin{array}{cccccccc} \bullet & \bullet & \bullet & \bullet & A_{64} & A_{65} & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & A_{74} & A_{75} & \bullet & \bullet \end{array} \right) \begin{pmatrix} x_4 \\ x_5 \end{pmatrix} \text{ temp} \leftarrow \begin{pmatrix} x_2 \\ x_3 \end{pmatrix}$$

31

PREMIÈRE ÉTAPE

$$P_0 \left( \begin{array}{cccccccc} A_{00} & A_{01} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ A_{10} & A_{11} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{array} \right) \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \text{ temp} \leftarrow \begin{pmatrix} x_6 \\ x_7 \end{pmatrix}$$

$$P_1 \left( \begin{array}{cccccccc} \bullet & \bullet & A_{22} & A_{23} & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & A_{32} & A_{33} & \bullet & \bullet & \bullet & \bullet \end{array} \right) \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} \text{ temp} \leftarrow \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$

$$P_2 \left( \begin{array}{cccccccc} \bullet & \bullet & \bullet & \bullet & A_{44} & A_{45} & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & A_{54} & A_{55} & \bullet & \bullet \end{array} \right) \begin{pmatrix} x_4 \\ x_5 \end{pmatrix} \text{ temp} \leftarrow \begin{pmatrix} x_2 \\ x_3 \end{pmatrix}$$

$$P_3 \left( \begin{array}{cccccccc} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{66} & A_{67} \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{76} & A_{77} \end{array} \right) \begin{pmatrix} x_6 \\ x_7 \end{pmatrix} \text{ temp} \leftarrow \begin{pmatrix} x_4 \\ x_5 \end{pmatrix}$$

TROISIÈME ÉTAPE

$$P_0 \left( \begin{array}{cccccccc} \bullet & \bullet & \bullet & \bullet & A_{04} & A_{05} & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & A_{14} & A_{15} & \bullet & \bullet \end{array} \right) \begin{pmatrix} x_4 \\ x_5 \end{pmatrix} \text{ temp} \leftarrow \begin{pmatrix} x_2 \\ x_3 \end{pmatrix}$$

$$P_1 \left( \begin{array}{cccccccc} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{26} & A_{27} \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{36} & A_{37} \end{array} \right) \begin{pmatrix} x_6 \\ x_7 \end{pmatrix} \text{ temp} \leftarrow \begin{pmatrix} x_4 \\ x_5 \end{pmatrix}$$

$$P_2 \left( \begin{array}{cccccccc} A_{40} & A_{41} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ A_{50} & A_{51} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{array} \right) \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \text{ temp} \leftarrow \begin{pmatrix} x_6 \\ x_7 \end{pmatrix}$$

$$P_3 \left( \begin{array}{cccccccc} \bullet & \bullet & A_{62} & A_{63} & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & A_{72} & A_{73} & \bullet & \bullet & \bullet & \bullet \end{array} \right) \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} \text{ temp} \leftarrow \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$



### QUATRIÈME ÉTAPE

$$P_0 \left( \begin{array}{cccccc} \bullet & \bullet & A_{02} & A_{03} & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & A_{12} & A_{13} & \bullet & \bullet & \bullet & \bullet \end{array} \right) \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} \text{temp} \leftarrow \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$

$$P_1 \left( \begin{array}{cccccc} \bullet & \bullet & \bullet & \bullet & A_{24} & A_{25} & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & A_{34} & A_{35} & \bullet & \bullet \end{array} \right) \begin{pmatrix} x_4 \\ x_5 \end{pmatrix} \text{temp} \leftarrow \begin{pmatrix} x_2 \\ x_3 \end{pmatrix}$$

$$P_2 \left( \begin{array}{cccccc} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{46} & A_{47} \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{56} & A_{57} \end{array} \right) \begin{pmatrix} x_6 \\ x_7 \end{pmatrix} \text{temp} \leftarrow \begin{pmatrix} x_4 \\ x_5 \end{pmatrix}$$

$$P_3 \left( \begin{array}{cccccc} A_{60} & A_{61} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ A_{70} & A_{71} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{array} \right) \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \text{temp} \leftarrow \begin{pmatrix} x_6 \\ x_7 \end{pmatrix}$$

33

### PROGRAMME

```
matrice-vecteur(A,x,y) {
  q = my_num();
  p = tot_proc_num();
  for (step=0;step<p;step++) {
    send(x,r);
    for (i=0;i<r;i++)
      for (j=0;j<r;j++)
        y[i] = y[i]+a[i,(q-step mod p)r+j]*x[j];
    receive(temp,r);
    x = temp;
  }
}
```

### PERFORMANCES

- en notant  $\tau_a$  le temps de calcul élémentaire,  $\tau_c$  le temps de communication élémentaire
- il y a  $p$  étapes identiques, chacune de temps égal le plus long entre le calcul local et le temps de communication:  $\max(r^2\tau_a, \beta + r\tau_c)$
- d'où temps total de  $p * \max(r^2\tau_a, \beta + r\tau_c)$
- quand  $n$  assez grand  $r^2\tau_a$  devient prépondérant, d'où asymptotiquement un temps de  $\frac{n^2}{p}\tau_a$ : efficacité 1!

(on aurait aussi pu procéder à un échange total de  $\mathbf{x}$  au début...)

35

### FACTORISATION LU

Problème: résolution d'un système linéaire dense  $Ax = b$  par triangulation de Gauss. Version séquentielle:

```
for (k=0;k<n-1;k++) {
  prep(k): for (i=k+1;i<n;i++)
    a[i,k]=a[i,k]/a[k,k];
  for (j=k+1;j<n;j++)
    update(k,j): for (i=k+1;i<n;i++)
      a[i,j]=a[i,j]-a[i,k]*a[k,j];
}
```

## DISTRIBUTION

- distribution des colonnes aux différents processeurs
- on suppose que cette distribution nous est donnée par une fonction `alloc` telle que `alloc(k)=q` veut dire que la  $k$ ème colonne est affectée à la mémoire locale de  $P_q$
- on utilise la fonction `broadcast`, pour faire en sorte qu'à l'étape  $k$ , le processeur qui possède la colonne  $k$  la diffuse à tous les autres

Voir poly pour version la plus générale.

37

### PROGRAMME - ICI `ALLOC(k)=k`

```

q = my_num();
p = tot_proc_num();
for (k=0;k<n-1;k++) {
  if (k == q) {
    prep(k): for (i=k+1;i<n;i++)
      buffer[i-k-1] = a[i,k]/a[k,k];
    broadcast(k,buffer,n-k);
  }
  else {
    receive(buffer,n-k);
    update(k,q): for (i=k+1;k<n;k++)
      a[i,q] = a[i,q]-buffer[i-k-1]*a[k,q]; }
}

```

## PROBLÈME DE L'ALLOCATION PAR BLOCS DE COLONNES

- le nombre de données varie au cours des étapes (de moins en moins)
- le volume de calcul n'est pas proportionnel au volume des données: quand un processeur a par exemple  $r$  colonnes consécutives, le dernier processeur a plus de calcul (que de données) par rapport au premier
- il faut donc une allocation qui réussisse à équilibrer le volume des données et du travail!
- équilibrage de charge à chaque étape de l'algorithme, et pas seulement global

39

## CAS DE L'ALLOCATION CYCLIQUE PAR COLONNES

$P_0$	$P_1$	$P_2$	$P_3$	$P_0$	$P_1$	$P_2$	$P_3$
$A_{00}$	$A_{01}$	$A_{02}$	$A_{03}$	$A_{04}$	$A_{05}$	$A_{06}$	$A_{07}$
$A_{10}$	$A_{11}$	$A_{12}$	$A_{13}$	$A_{14}$	$A_{15}$	$A_{16}$	$A_{17}$
$A_{20}$	$A_{21}$	$A_{22}$	$A_{23}$	$A_{24}$	$A_{25}$	$A_{26}$	$A_{27}$
$A_{30}$	$A_{31}$	$A_{32}$	$A_{33}$	$A_{34}$	$A_{35}$	$A_{36}$	$A_{37}$
$A_{40}$	$A_{41}$	$A_{42}$	$A_{43}$	$A_{44}$	$A_{45}$	$A_{46}$	$A_{47}$
$A_{50}$	$A_{51}$	$A_{52}$	$A_{53}$	$A_{54}$	$A_{55}$	$A_{56}$	$A_{57}$
$A_{60}$	$A_{61}$	$A_{62}$	$A_{63}$	$A_{64}$	$A_{65}$	$A_{66}$	$A_{67}$
$A_{70}$	$A_{71}$	$A_{72}$	$A_{73}$	$A_{74}$	$A_{75}$	$A_{76}$	$A_{77}$

ALLOCATION CYCLIQUE -  $\kappa=0$

$P_0 : p_0; b$	$P_1$	$P_2$	$P_3$	$P_0$	$P_1$	$P_2$	$P_3$
$U_{00}$	$A_{01}$	$A_{02}$	$A_{03}$	$A_{04}$	$A_{05}$	$A_{06}$	$A_{07}$
$L_{10}$	$A_{11}$	$A_{12}$	$A_{13}$	$A_{14}$	$A_{15}$	$A_{16}$	$A_{17}$
$L_{20}$	$A_{21}$	$A_{22}$	$A_{23}$	$A_{24}$	$A_{25}$	$A_{26}$	$A_{27}$
$L_{30}$	$A_{31}$	$A_{32}$	$A_{33}$	$A_{34}$	$A_{35}$	$A_{36}$	$A_{37}$
$L_{40}$	$A_{41}$	$A_{42}$	$A_{43}$	$A_{44}$	$A_{45}$	$A_{46}$	$A_{47}$
$L_{50}$	$A_{51}$	$A_{52}$	$A_{53}$	$A_{54}$	$A_{55}$	$A_{56}$	$A_{57}$
$L_{60}$	$A_{61}$	$A_{62}$	$A_{63}$	$A_{64}$	$A_{65}$	$A_{66}$	$A_{67}$
$L_{70}$	$A_{71}$	$A_{72}$	$A_{73}$	$A_{74}$	$A_{75}$	$A_{76}$	$A_{77}$

41

PUIS...

$P_0$	$P_1$	$P_2$	$P_3$	$P_0$	$P_1 : r; u_{0,5}$	$P_2 : r; u_{0,6}$	$P_3 : r; u_{0,7}$
$U_{00}$	$U_{01}$	$U_{02}$	$U_{03}$	$U_{04}$	$U_{05}$	$U_{06}$	$U_{07}$
$L_{10}$	$A'_{11}$	$A'_{12}$	$A'_{13}$	$A'_{14}$	$A'_{15}$	$A'_{16}$	$A'_{17}$
$L_{20}$	$A'_{21}$	$A'_{22}$	$A'_{23}$	$A'_{24}$	$A'_{25}$	$A'_{26}$	$A'_{27}$
$L_{30}$	$A'_{31}$	$A'_{32}$	$A'_{33}$	$A'_{34}$	$A'_{35}$	$A'_{36}$	$A'_{37}$
$L_{40}$	$A'_{41}$	$A'_{42}$	$A'_{43}$	$A'_{44}$	$A'_{45}$	$A'_{46}$	$A'_{47}$
$L_{50}$	$A'_{51}$	$A'_{52}$	$A'_{53}$	$A'_{54}$	$A'_{55}$	$A'_{56}$	$A'_{57}$
$L_{60}$	$A'_{61}$	$A'_{62}$	$A'_{63}$	$A'_{64}$	$A'_{65}$	$A'_{66}$	$A'_{67}$
$L_{70}$	$A'_{71}$	$A'_{72}$	$A'_{73}$	$A'_{74}$	$A'_{75}$	$A'_{76}$	$A'_{77}$

43

ALLOCATION CYCLIQUE -  $\kappa=0$

$P_0$	$P_1 : r; u_{0,1}$	$P_2 : r; u_{0,2}$	$P_3 : r; u_{0,3}$	$P_0 : u_{0,4}$	$P_1$	$P_2$	$P_3$
$U_{00}$	$U_{01}$	$U_{02}$	$U_{03}$	$U_{04}$	$A_{05}$	$A_{06}$	$A_{07}$
$L_{10}$	$A'_{11}$	$A'_{12}$	$A'_{13}$	$A'_{14}$	$A_{15}$	$A_{16}$	$A_{17}$
$L_{20}$	$A'_{21}$	$A'_{22}$	$A'_{23}$	$A'_{24}$	$A_{25}$	$A_{26}$	$A_{27}$
$L_{30}$	$A'_{31}$	$A'_{32}$	$A'_{33}$	$A'_{34}$	$A_{35}$	$A_{36}$	$A_{37}$
$L_{40}$	$A'_{41}$	$A'_{42}$	$A'_{43}$	$A'_{44}$	$A_{45}$	$A_{46}$	$A_{47}$
$L_{50}$	$A'_{51}$	$A'_{52}$	$A'_{53}$	$A'_{54}$	$A_{55}$	$A_{56}$	$A_{57}$
$L_{60}$	$A'_{61}$	$A'_{62}$	$A'_{63}$	$A'_{64}$	$A_{65}$	$A_{66}$	$A_{67}$
$L_{70}$	$A'_{71}$	$A'_{72}$	$A'_{73}$	$A'_{74}$	$A_{75}$	$A_{76}$	$A_{77}$

ALLOCATION CYCLIQUE -  $\kappa=1$

$P_0$	$P_1 : p_1; b$	$P_2$	$P_3$	$P_0$	$P_1$	$P_2$	$P_3$
$U_{00}$	$U_{01}$	$U_{02}$	$U_{03}$	$U_{04}$	$U_{05}$	$U_{06}$	$U_{07}$
$L_{10}$	$U_{11}$	$A'_{12}$	$A'_{13}$	$A'_{14}$	$A'_{15}$	$A'_{16}$	$A'_{17}$
$L_{20}$	$L_{21}$	$A'_{22}$	$A'_{23}$	$A'_{24}$	$A'_{25}$	$A'_{26}$	$A'_{27}$
$L_{30}$	$L_{31}$	$A'_{32}$	$A'_{33}$	$A'_{34}$	$A'_{35}$	$A'_{36}$	$A'_{37}$
$L_{40}$	$L_{41}$	$A'_{42}$	$A'_{43}$	$A'_{44}$	$A'_{45}$	$A'_{46}$	$A'_{47}$
$L_{50}$	$L_{51}$	$A'_{52}$	$A'_{53}$	$A'_{54}$	$A'_{55}$	$A'_{56}$	$A'_{57}$
$L_{60}$	$L_{61}$	$A'_{62}$	$A'_{63}$	$A'_{64}$	$A'_{65}$	$A'_{66}$	$A'_{67}$
$L_{70}$	$L_{71}$	$A'_{72}$	$A'_{73}$	$A'_{74}$	$A'_{75}$	$A'_{76}$	$A'_{77}$

ALLOCATION CYCLIQUE -  $\kappa=1$

$P_0$	$P_1$	$P_2 : r; u_{1,2}$	$P_3 : r; u_{1,3}$	$P_0 : r; u_{1,4}$	$P_1$	$P_2$	$P_3$
$U_{00}$	$U_{01}$	$U_{02}$	$U_{03}$	$U_{04}$	$U_{05}$	$U_{06}$	$U_{07}$
$L_{10}$	$U_{11}$	$U_{12}$	$U_{13}$	$U_{14}$	$A'_{15}$	$A'_{16}$	$A'_{17}$
$L_{20}$	$L_{21}$	$A''_{22}$	$A''_{23}$	$A''_{24}$	$A'_{25}$	$A'_{26}$	$A'_{27}$
$L_{30}$	$L_{31}$	$A''_{32}$	$A''_{33}$	$A''_{34}$	$A'_{35}$	$A'_{36}$	$A'_{37}$
$L_{40}$	$L_{41}$	$A''_{42}$	$A''_{43}$	$A''_{44}$	$A'_{45}$	$A'_{46}$	$A'_{47}$
$L_{50}$	$L_{51}$	$A''_{52}$	$A''_{53}$	$A''_{54}$	$A'_{55}$	$A'_{56}$	$A'_{57}$
$L_{60}$	$L_{61}$	$A''_{62}$	$A''_{63}$	$A''_{64}$	$A'_{65}$	$A'_{66}$	$A'_{67}$
$L_{70}$	$L_{71}$	$A''_{72}$	$A''_{73}$	$A''_{74}$	$A'_{75}$	$A'_{76}$	$A'_{77}$

45

CAS DE L'ALLOCATION 1 COLONNE 1 PROCESSEUR -  $P=N$

Ici,  $\text{alloc}(k)=k$

- Coût de la mise à jour (**update**) de la colonne  $j$  par le processeur  $j$ :
  - à toutes les étapes  $k = 0$  à  $k = n - 1$
  - un coût de  $n - k - 1$  pour l'étape  $k$  (éléments en position  $k + 1$  à  $n - 1$ )
  - d'où un coût total de

$$t = \sum_{k=0}^{n-1} (n - k - 1)\tau_a = \frac{n(n-1)}{2}\tau_a$$

TEMPS DE CALCUL

- Le chemin critique d'exécution est:  
 $prep_0(0) \rightarrow update_1(0, 1), prep_1(1) \rightarrow update_2(1, 2), prep_2(2) \rightarrow \dots$   
 ( $\rightarrow$  = communication vers proc. voisin)
- Comme si on faisait environ  $r$  fois le travail quand allocation cyclique pour  $r = \frac{n}{p}$  processeurs
- Remarque: recouvrement des communications, mais pas communication/calcul!

47

TEMPS DE CALCUL

- $n\beta + \frac{n^2}{2}\tau_c + O(1)$  pour les  $n - 1$  communications (transportant de l'ordre de  $n^2$  données)
- $\frac{n^2}{2}\tau_a + O(1)$  pour les **prep**
- Pour l'**update** des  $r$  colonnes sur le processeur  $j \bmod p$ , en parallèle sur tous les processeurs, environ  $r \frac{n(n-1)}{2}$
- D'où un coût de l'ordre de  $\frac{n^3}{2p}$  pour les **update** des  $p$  processeurs: terme dominant si  $p \ll n$  et efficacité excellente asymptotiquement (pour  $n$  grand)

```

q = my_num();
p = tot_proc_num();
l = 0;
for (k=0;k<n-1;k++) {
  if (k == q mod p) {
    prep(k): for (i=k+1;i<n;i++)
      buffer[i-k-1] = a[i,1]/a[k,1];
    l++; send(buffer,n-k); }
  else { receive(buffer,n-k);
    if (q != k-1 mod p) send(buffer,n-k); }
  for (j=1;j<r;j++)
    update(k,j): for (i=k+1;k<n;k++)
      a[i,j] = a[i,j]-buffer[i-k-1]*a[k,j]; }

```

49

DÉFAUT...

Sur  $P_1$ :

- Etape  $k = 0$ :  $P_1$  reçoit la colonne pivot 0 de  $P_0$
- $P_1$  l'envoie à  $P_2$
- Fait `update(0, j)` pour toutes les colonnes  $j$  qui lui appartiennent, c.-a-d.  $j = 1 \bmod p$
- Etape  $k = 1$ : fait `prep(1)`
- Envoie la colonne pivot 1 à  $P_2$
- Fait `update(1, j)` pour toutes les colonnes  $j$  qui lui appartiennent, c.-a-d.  $j = 1 \bmod p$
- etc.

$P_0$	$P_1$	$P_2$	$P_3$
<i>prep(0)</i>			
<i>send(0)</i>	<i>receive(0)</i>		
<i>update(0, 4)</i>	<i>send(0)</i>	<i>receive(0)</i>	
<i>update(0, 8)</i>	<i>update(0, 1)</i>	<i>send(0)</i>	<i>receive(0)</i>
<i>update(0, 12)</i>	<i>update(0, 5)</i>	<i>update(0, 2)</i>	<i>update(0, 3)</i>
	<i>update(0, 9)</i>	<i>update(0, 6)</i>	<i>update(0, 7)</i>
	<i>update(0, 13)</i>	<i>update(0, 10)</i>	<i>update(0, 11)</i>
	<i>prep(1)</i>	<i>update(0, 14)</i>	<i>update(0, 15)</i>
	<i>send(1)</i>	<i>receive(1)</i>	
	<i>update(1, 5)</i>	<i>send(1)</i>	<i>receive(1)</i>
<i>receive(1)</i>	<i>update(1, 9)</i>	<i>update(1, 2)</i>	<i>send(1)</i>
<i>update(1, 4)</i>	<i>update(1, 13)</i>	<i>update(1, 6)</i>	<i>update(1, 3)</i>
<i>update(1, 8)</i>		<i>update(1, 10)</i>	<i>update(1, 7)</i>
<i>update(1, 12)</i>		<i>update(1, 14)</i>	<i>update(1, 11)</i>
...	...	...	...

51

DÉFAUT...

$P_1$  aurait pu faire:

- `update(0, 1)`
- `prep(1)`
- Envoi vers  $P_2$
- `update(0, j)` pour  $j = 1 \bmod p$  et  $j > 1$
- etc.

$P_0$	$P_1$	$P_2$	$P_3$
$prep(0)$			
$send(0)  up(0,4)$	$receive(0)$		
$up(0,8)$	$send(0)  up(0,1)$	$receive(0)$	
$up(0,12)$	$prep(1)$	$send(0)  up(0,2)$	$receive(0)$
	$send(1)  up(0,5)$	$receive(1)  up(0,6)$	$up(0,3)$
	$up(0,9)$	$send(1)  up(0,10)$	$receive(1)  up(0,7)$
$receive(1)$	$up(0,13)$	$up(0,14)$	$send(1)  up(0,11)$
$up(1,4)$	$up(1,5)$	$up(1,2)$	$up(0,15)$
$up(1,8)$	$up(1,9)$	$prep(2)$	$up(1,3)$
$up(1,12)$	$up(1,13)$	$send(2)  up(1,6)$	$receive(2)  up(1,7)$
$receive(2)$		$up(1,10)$	$send(2)  up(1,11)$
$send(2)  up(2,4)$	$receive(2)$	$up(1,14)$	$up(1,15)$
...	...	...	...