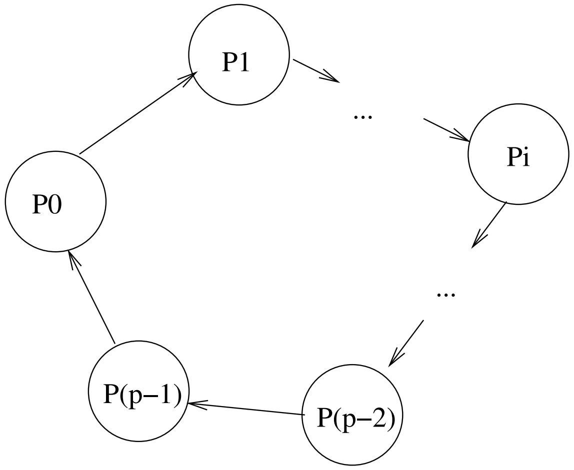
Parallélisme

Eric Goubault
Commissariat à l'Energie Atomique
Saclay

Algorithmique sur anneau de processeurs

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

Architecture



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 `proc_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 `proc_num()-1[p]` par `receive(adr,L)`

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

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: aucune bloquante (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:

- β est le coût d'initialisation (latence)
- τ (débit) mesure la vitesse de transmission en régime permanent

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

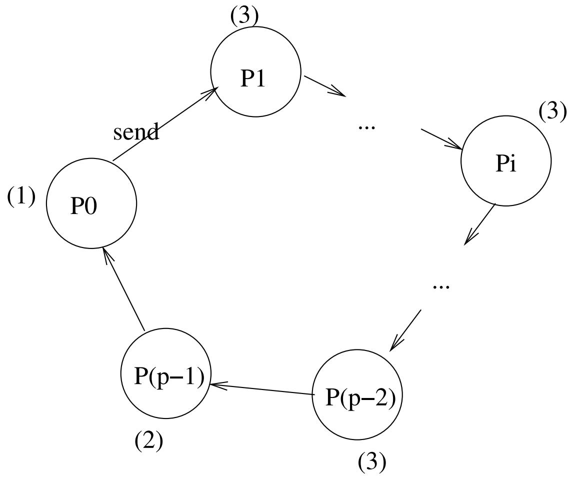
Problème élémentaire: la diffusion

- 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 faon efficace dans la plupart des librairies de communications (PVM, MPI etc.).

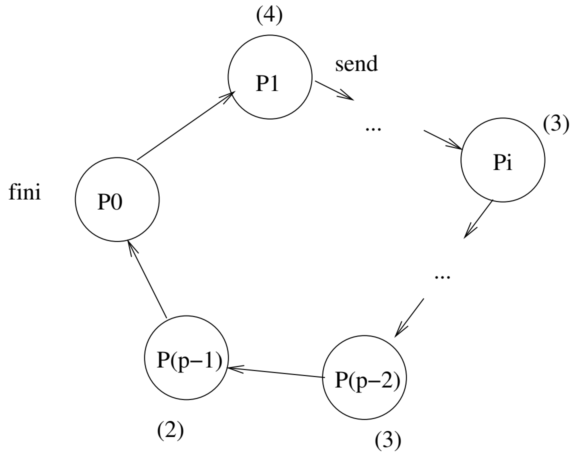
Implémentation (`receive` *bloquant*)

```
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);
    }
}
```

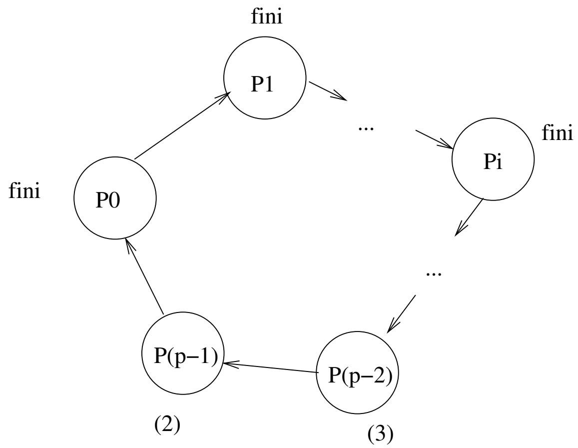

Exécution - temps 0 et `k=0`



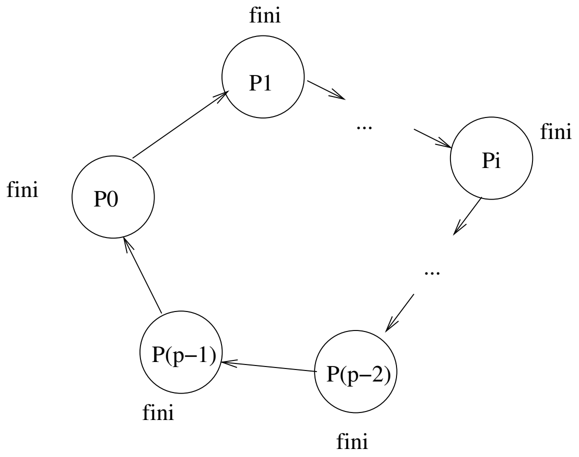
Exécution - temps $\beta + L\tau$



Exécution - temps $k(\beta + L\tau)$ $(k < p-1)$



Exécution - temps $(p-1)(\beta + L\tau)$



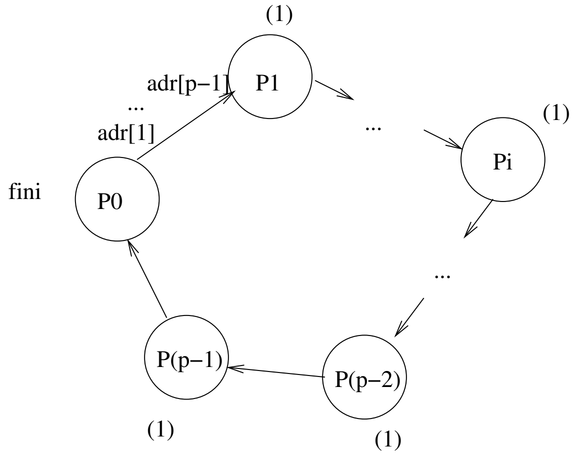
Diffusion personnalisée

- `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!

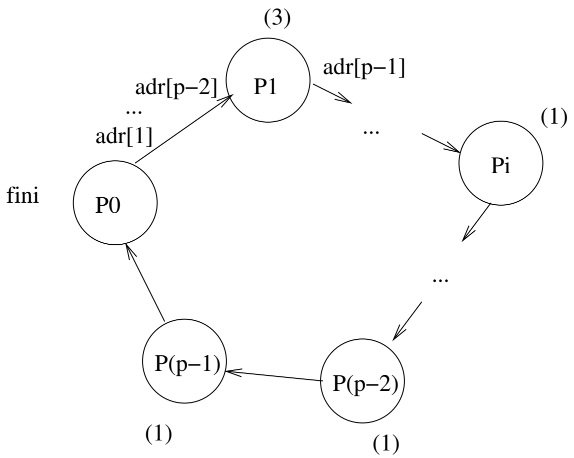
Programme

```
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; } }
```

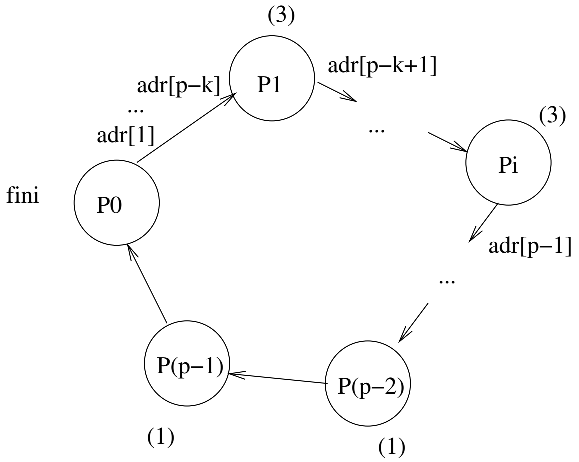

Exécution - temps 0 et `k=0`



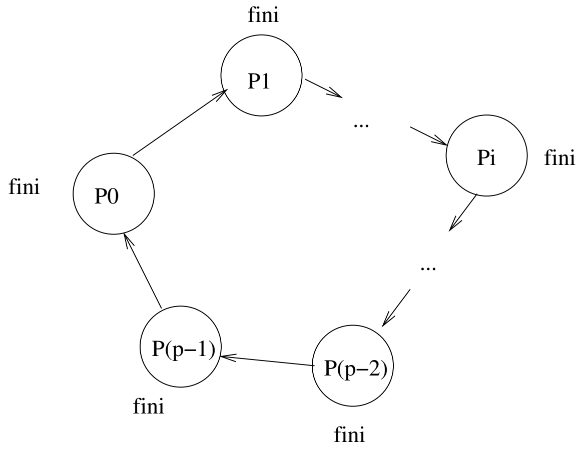
Exécution - temps $\beta + L\tau$



Exécution - temps $i(\beta + L\tau)$



Exécution - temps $(p-1)(\beta + L\tau)$



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);
  }
}
```


Diffusion pipelinée

Le 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'emetteur envoie successivement les r morceaux, avec recouvrement partiel des communications
- au début ces morceaux de messages sont dans `adr[1]`,...,`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); } } }
```


Temps d'exécution

- le premier morceau de longueur `L/r` du message sera arrivé au dernier processeur `k-1 mod 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!

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];
```


Principe de la distribution

Distribuer le calcul des produits scalaires aux processeurs:

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

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

Principe du calcul -distribution initiale des données

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

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

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

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

Première étape

$$P_0 \begin{pmatrix} A_{00} & A_{01} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ A_{10} & A_{11} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_6 \\ x_7 \end{pmatrix}$$

$$P_1 \begin{pmatrix} \bullet & \bullet & A_{22} & A_{23} & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & A_{32} & A_{33} & \bullet & \bullet & \bullet & \bullet \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$

$$P_2 \begin{pmatrix} \bullet & \bullet & \bullet & \bullet & A_{44} & A_{45} & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & A_{54} & A_{55} & \bullet & \bullet \end{pmatrix} \begin{pmatrix} x_4 \\ x_5 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_2 \\ x_3 \end{pmatrix}$$

$$P_3 \begin{pmatrix} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{66} & A_{67} \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{76} & A_{77} \end{pmatrix} \begin{pmatrix} x_6 \\ x_7 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_4 \\ x_5 \end{pmatrix}$$

Deuxième étape

$$P_0 \begin{pmatrix} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{06} & A_{07} \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{16} & A_{17} \end{pmatrix} \begin{pmatrix} x_6 \\ x_7 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_4 \\ x_5 \end{pmatrix}$$

$$P_1 \begin{pmatrix} A_{20} & A_{21} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ A_{30} & A_{31} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_6 \\ x_7 \end{pmatrix}$$

$$P_2 \begin{pmatrix} \bullet & \bullet & A_{42} & A_{43} & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & A_{52} & A_{53} & \bullet & \bullet & \bullet & \bullet \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$

$$P_3 \begin{pmatrix} \bullet & \bullet & \bullet & \bullet & A_{64} & A_{65} & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & A_{74} & A_{75} & \bullet & \bullet \end{pmatrix} \begin{pmatrix} x_4 \\ x_5 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_2 \\ x_3 \end{pmatrix}$$

Troisième étape

$$P_0 \begin{pmatrix} \bullet & \bullet & \bullet & \bullet & A_{04} & A_{05} & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & A_{14} & A_{15} & \bullet & \bullet \end{pmatrix} \begin{pmatrix} x_4 \\ x_5 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_2 \\ x_3 \end{pmatrix}$$

$$P_1 \begin{pmatrix} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{26} & A_{27} \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{36} & A_{37} \end{pmatrix} \begin{pmatrix} x_6 \\ x_7 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_4 \\ x_5 \end{pmatrix}$$

$$P_2 \begin{pmatrix} A_{40} & A_{41} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ A_{50} & A_{51} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_6 \\ x_7 \end{pmatrix}$$

$$P_3 \begin{pmatrix} \bullet & \bullet & A_{62} & A_{63} & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & A_{72} & A_{73} & \bullet & \bullet & \bullet & \bullet \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$

Quatrième étape

$$P_0 \begin{pmatrix} \bullet & \bullet & A_{02} & A_{03} & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & A_{12} & A_{13} & \bullet & \bullet & \bullet & \bullet \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$

$$P_1 \begin{pmatrix} \bullet & \bullet & \bullet & \bullet & A_{24} & A_{25} & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & A_{34} & A_{35} & \bullet & \bullet \end{pmatrix} \begin{pmatrix} x_4 \\ x_5 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_2 \\ x_3 \end{pmatrix}$$

$$P_2 \begin{pmatrix} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{46} & A_{47} \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & A_{56} & A_{57} \end{pmatrix} \begin{pmatrix} x_6 \\ x_7 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_4 \\ x_5 \end{pmatrix}$$

$$P_3 \begin{pmatrix} A_{60} & A_{61} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ A_{70} & A_{71} & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \; temp \leftarrow \begin{pmatrix} x_6 \\ x_7 \end{pmatrix}$$

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 τ_a le temps de calcul élémentaire, τ_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 asympotiquement un temps de $\frac{n^2}{p}\tau_a$: efficacité 1!

(on aurait aussi pu procéder à un échange total de `x` au début...)

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;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;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 $k$ième 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.

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;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;k<n;k++)
        a[i,q] = a[i,q]+buffer[i-k]*a[k,q]; }
}
```


Difficultés de l'algorithme

- 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` lignes consécutives, le dernier processeur a moins 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

Cas de l'allocation cyclique par lignes

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

Allocation cyclique - κ=0

$$
\begin{array}{c l}
P_0 : prep(0); broadcast() & (-1 \;\; A'_{01} \;\; A'_{02} \;\; A'_{03} \;\; A'_{04} \;\; A'_{05} \;\; A'_{06} \;\; A'_{07}) \\
P_1 & (A_{10} \;\; A_{11} \;\; A_{12} \;\; A_{13} \;\; A_{14} \;\; A_{15} \;\; A_{16} \;\; A_{17}) \\
P_2 & (A_{20} \;\; A_{21} \;\; A_{22} \;\; A_{23} \;\; A_{24} \;\; A_{25} \;\; A_{26} \;\; A_{27}) \\
P_3 & (A_{30} \;\; A_{31} \;\; A_{32} \;\; A_{33} \;\; A_{34} \;\; A_{35} \;\; A_{36} \;\; A_{37}) \\
\hline
P_0 & (A_{40} \;\; A_{41} \;\; A_{42} \;\; A_{43} \;\; A_{44} \;\; A_{45} \;\; A_{46} \;\; A_{47}) \\
P_1 & (A_{50} \;\; A_{51} \;\; A_{52} \;\; A_{53} \;\; A_{54} \;\; A_{55} \;\; A_{56} \;\; A_{57}) \\
P_2 & (A_{60} \;\; A_{61} \;\; A_{62} \;\; A_{63} \;\; A_{64} \;\; A_{65} \;\; A_{66} \;\; A_{67}) \\
P_3 & (A_{70} \;\; A_{71} \;\; A_{72} \;\; A_{73} \;\; A_{74} \;\; A_{75} \;\; A_{76} \;\; A_{77})
\end{array}
$$

Allocation cyclique - κ=0

$$
\begin{array}{c l}
P_0 & (-1 \;\; A'_{01} \;\; A'_{02} \;\; A'_{03} \;\; A'_{04} \;\; A'_{05} \;\; A'_{06} \;\; A'_{07}) \\
P_1 : receive(); update(0,1) & (0 \;\; A'_{11} \;\; A'_{12} \;\; A'_{13} \;\; A'_{14} \;\; A'_{15} \;\; A'_{16} \;\; A'_{17}) \\
P_2 : receive(); update(0,2) & (0 \;\; A'_{21} \;\; A'_{22} \;\; A'_{23} \;\; A'_{24} \;\; A'_{25} \;\; A'_{26} \;\; A'_{27}) \\
P_3 : receive(); update(0,3) & (0 \;\; A'_{31} \;\; A'_{32} \;\; A'_{33} \;\; A'_{34} \;\; A'_{35} \;\; A'_{36} \;\; A'_{37}) \\
\hline
P_0 : update(0,4) & (0 \;\; A'_{41} \;\; A'_{42} \;\; A'_{43} \;\; A'_{44} \;\; A'_{45} \;\; A'_{46} \;\; A'_{47}) \\
P_1 & (A_{50} \;\; A_{51} \;\; A_{52} \;\; A_{53} \;\; A_{54} \;\; A_{55} \;\; A_{56} \;\; A_{57}) \\
P_2 & (A_{60} \;\; A_{61} \;\; A_{62} \;\; A_{63} \;\; A_{64} \;\; A_{65} \;\; A_{66} \;\; A_{67}) \\
P_3 & (A_{70} \;\; A_{71} \;\; A_{72} \;\; A_{73} \;\; A_{74} \;\; A_{75} \;\; A_{76} \;\; A_{77})
\end{array}
$$

Puis...

$$
\begin{array}{c l}
P_0 & (-1 \;\; A'_{01} \;\; A'_{02} \;\; A'_{03} \;\; A'_{04} \;\; A'_{05} \;\; A'_{06} \;\; A'_{07}) \\
P_1 & (0 \;\; A'_{11} \;\; A'_{12} \;\; A'_{13} \;\; A'_{14} \;\; A'_{15} \;\; A'_{16} \;\; A'_{17}) \\
P_2 & (0 \;\; A'_{21} \;\; A'_{22} \;\; A'_{23} \;\; A'_{24} \;\; A'_{25} \;\; A'_{26} \;\; A'_{27}) \\
P_3 & (0 \;\; A'_{31} \;\; A'_{32} \;\; A'_{33} \;\; A'_{34} \;\; A'_{35} \;\; A'_{36} \;\; A'_{37}) \\
\hline
P_0 & (0 \;\; A'_{41} \;\; A'_{42} \;\; A'_{43} \;\; A'_{44} \;\; A'_{45} \;\; A'_{46} \;\; A'_{47}) \\
P_1 : receive(); update(0,5) & (0 \;\; A'_{51} \;\; A'_{52} \;\; A'_{53} \;\; A'_{54} \;\; A'_{55} \;\; A'_{56} \;\; A'_{57}) \\
P_2 : receive(); update(0,6) & (0 \;\; A'_{61} \;\; A'_{62} \;\; A'_{63} \;\; A'_{64} \;\; A'_{65} \;\; A'_{66} \;\; A'_{67}) \\
P_3 : receive(); update(0,7) & (0 \;\; A'_{71} \;\; A'_{72} \;\; A'_{73} \;\; A'_{74} \;\; A'_{75} \;\; A'_{76} \;\; A'_{77})
\end{array}
$$

Allocation cyclique - κ=1

$$
\begin{array}{c l}
P_0 & (-1 \;\; A'_{01} \;\; A'_{02} \;\; A'_{03} \;\; A'_{04} \;\; A'_{05} \;\; A'_{06} \;\; A'_{07}) \\
P_1 : prep(1); broadcast() & (0 \;\; -1 \;\; A''_{12} \;\; A''_{13} \;\; A''_{14} \;\; A''_{15} \;\; A''_{16} \;\; A''_{17}) \\
P_2 & (0 \;\; A'_{21} \;\; A'_{22} \;\; A'_{23} \;\; A'_{24} \;\; A'_{25} \;\; A'_{26} \;\; A'_{27}) \\
P_3 & (0 \;\; A'_{31} \;\; A'_{32} \;\; A'_{33} \;\; A'_{34} \;\; A'_{35} \;\; A'_{36} \;\; A'_{37}) \\
\hline
P_0 & (0 \;\; A'_{41} \;\; A'_{42} \;\; A'_{43} \;\; A'_{44} \;\; A'_{45} \;\; A'_{46} \;\; A'_{47}) \\
P_1 & (0 \;\; A'_{51} \;\; A'_{52} \;\; A'_{53} \;\; A'_{54} \;\; A'_{55} \;\; A'_{56} \;\; A'_{57}) \\
P_2 & (0 \;\; A'_{61} \;\; A'_{62} \;\; A'_{63} \;\; A'_{64} \;\; A'_{65} \;\; A'_{66} \;\; A'_{67}) \\
P_3 & (0 \;\; A'_{71} \;\; A'_{72} \;\; A'_{73} \;\; A'_{74} \;\; A'_{75} \;\; A'_{76} \;\; A'_{77})
\end{array}
$$

ALLOCATION CYCLIQUE - K=1

$$\begin{array}{rl}
P_0 & \begin{pmatrix} -1 & A'_{01} & A'_{02} & A'_{03} & A'_{04} & A'_{05} & A'_{06} & A'_{07} \end{pmatrix} \\
P_1 & \begin{pmatrix} 0 & -1 & A''_{12} & A''_{13} & A''_{14} & A''_{15} & A''_{16} & A''_{17} \end{pmatrix} \\
P_2 : receive(); update(1,2) & \begin{pmatrix} 0 & 0 & A''_{22} & A''_{23} & A''_{24} & A''_{25} & A''_{26} & A''_{27} \end{pmatrix} \\
P_3 : receive(); update(1,3) & \begin{pmatrix} 0 & 0 & A''_{32} & A''_{33} & A''_{34} & A''_{35} & A''_{36} & A''_{37} \end{pmatrix} \\ \hline
P_0 : receive(); update(1,4) & \begin{pmatrix} 0 & 0 & A''_{42} & A''_{43} & A''_{44} & A''_{45} & A''_{46} & A''_{47} \end{pmatrix} \\
P_1 & \begin{pmatrix} 0 & A'_{51} & A'_{52} & A'_{53} & A'_{54} & A'_{55} & A'_{56} & A'_{57} \end{pmatrix} \\
P_2 & \begin{pmatrix} 0 & A'_{61} & A'_{62} & A'_{63} & A'_{64} & A'_{65} & A'_{66} & A'_{67} \end{pmatrix} \\
P_3 & \begin{pmatrix} 0 & A'_{71} & A'_{72} & A'_{73} & A'_{74} & A'_{75} & A'_{76} & A'_{77} \end{pmatrix}
\end{array}$$

CAS DE L'ALLOCATION 1 LIGNE 1 PROCESSEUR - P=N

Ici, `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

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

TEMPS DE CALCUL

- Le chemin critique d'exécution est:

  $prep_0(0) \to update_1(0,1), prep_1(1) \to update_2(1,2), prep_2(2) \to \ldots$
- 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!

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 $j \ mod \ p$, 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 $p << n$ et efficacité excellente asymptotiquement (pour n grand)

Sur anneau: recouvrement communication/calcul

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


Défaut...

Sur P_1:

- Etape $k = 0$: P_1 reçoit la colonne pivot 0 de P_0
- P_1 l'envoit à P_2
- Fait `update(0,j)` pour toutes les colonnes j qui lui appartiennent, cad $j = 1 \; mod \; p$
- 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, cad $j = 1 \; mod \; p$
- etc.

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

Défaut...

P_1 aurait pu faire:

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

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