Réseaux d'interconnexion

1 Anneaux - rotations de Givens

Pour trianguliariser une matrice A d'ordre n de façon numériquement stable, on peut utiliser les rotations de Givens. L'opération de base $\textsf{Rot}(i,j,k)$ consiste à combiner les deux lignes i et j, qui doivent toutes deux commencer par $k-1$ zéros, pour annuler l'élément en position (j,k) :

$$\begin{pmatrix} 0 & \dots & 0 & \mathbf{a'_{i,k}} & a'_{i,k+1} & \dots & a'_{i,n} \\ 0 & \dots & 0 & \mathbf{0} & a'_{j,k+1} & \dots & a'_{j,n} \end{pmatrix} \leftarrow \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} 0 & \dots & 0 & \mathbf{a_{i,k}} & a_{i,k+1} & \dots & a_{i,n} \\ 0 & \dots & 0 & \mathbf{a_{j,k}} & a_{j,k+1} & \dots & a_{j,n} \end{pmatrix}$$

3p Nous laissons au lecteur le soin de déterminer l'angle θ permettant d'effectuer cette opération. :-)

L'algorithme séquentiel peut s'écrire :

```
Givens(A)
1: Pour k = 1 to n − 1 :
2:     Pour i = n downto k + 1 step −1 :
3:         Rot(i − 1, i, k)
```

On considère qu'une rotation $\textsf{Rot}(i,j,k)$ s'exécute en temps unité, indépendamment de k.

▷ **Question 1** *Mettre en œuvre cet algorithme sur un réseau linéaire de n processeurs.*

▷ **Question 2** *Mettre en œuvre cet algorithme sur un réseau linéaire comportant seulement $\lfloor \frac{n}{2} \rfloor$ processeurs.*

2 Tores, hypercubes et arbres binaires

▷ **Question 3** *Quelles différences peut-on trouver entre un hypercube de dimension 6 et un tore 3D de taille $4 \times 4 \times 4$?*

▷ **Question 4** *Peut-on plonger un arbre binaire complet à $2^n - 1$ sommets dans une grille 2D de taille $r \times r$ à déterminer ?*

3 Cycles connectés en cube

Un réseau $CCC(m)$ est obtenu en remplaçant chaque processeur d'un hypercube de dimension m par un anneau de m processeurs, et en connectant chaque processeur de l'anneau dans une dimension de l'hypercube (voir figure 1).

▷ **Question 5** *Quel est le nombre de processeurs de $CCC(m)$? Donner une définition formelle de $CCC(m)$ et une majoration simple de son diamètre.*

▷ **Question 6** *Montrer que le diamètre de $CCC(m)$ est exactement $D = 2m - 2 + \lfloor \frac{m}{2} \rfloor$ si $m > 3$, et est égal à 6 si $m = 3$.*

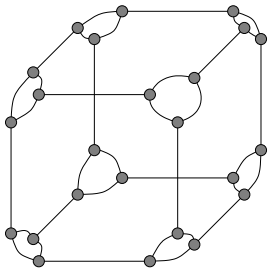

FIG. 1 – Cube connecté en cycle de taille 3.

4 Transposition d'une matrice

On veut concevoir un algorithme parallèle pour la transposition d'une matrice $n \times n$. On suppose la matrice stockée de manière distribuée dans les processeurs. On supposera que les différents liens de communication sont bidirectionnels et peuvent être utilisés de manière simultanée.

▷ **Question 7** *Proposer une solution sur un anneau de p processeurs et donner sa complexité (on suppose que la distribution est monodimensionnelle).*

▷ **Question 8** *Proposer une solution sur une grille torique de $p = q \times q$ processeurs et donner sa complexité (on suppose que la distribution est bidimensionnelle).*

▷ **Question 9** *Proposer une solution sur un hypercube de $p = 2^{2k}$ processeurs et donner sa complexité (on suppose que la distribution est bidimensionnelle).*

5 Réponses aux exercices

▷ Question 1, page 1

La mise en œuvre avec n processeurs numérotés P_1 à P_n est aisée : le processeur P_k va être responsable de toutes les rotations $\textsc{Rot}(i-1,i,k)$. On a en entrée du réseau (où $\boxed{i}$ représente la ligne i de la matrice) :

$$\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{5}\boxed{6}\boxed{7}\boxed{8} \rightarrow P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_5 \rightarrow P_6 \rightarrow P_7 \rightarrow P_8$$

La première ligne qui arrive dans un processeur s'y installe pour un top. Quand une autre ligne arrive, elle est combinée avec la ligne en mémoire, puis prend sa place. La ligne qui était en mémoire poursuit sa route vers la droite. On obtient le schéma d'exécution suivant pour $n = 8$:

Top	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
$t=1$	$\boxed{8}$							
$t=2$	Rot(7,8,1)							
$t=3$	Rot(6,7,1)	$\boxed{8}$						
$t=4$	Rot(5,6,1)	Rot(7,8,2)						
$t=5$	Rot(4,5,1)	Rot(6,7,2)	$\boxed{8}$					
$t=6$	Rot(3,4,1)	Rot(5,6,2)	Rot(7,8,3)					
$t=7$	Rot(2,3,1)	Rot(4,5,2)	Rot(6,7,3)	$\boxed{8}$				
$t=8$	Rot(1,2,1)	Rot(3,4,2)	Rot(5,6,3)	Rot(7,8,4)				
$t=9$	$\boxed{1}$	Rot(2,3,2)	Rot(4,5,3)	Rot(6,7,4)	$\boxed{8}$			
$t=10$	$\boxed{1}$	$\boxed{2}$	Rot(3,4,3)	Rot(5,6,4)	Rot(7,8,5)			
$t=11$	$\boxed{1}$	$\boxed{2}$	$\boxed{3}$	Rot(4,5,4)	Rot(6,7,5)	$\boxed{8}$		
$t=12$	$\boxed{1}$	$\boxed{2}$	$\boxed{3}$	$\boxed{4}$	Rot(5,6,5)	Rot(7,8,6)		
$t=13$	$\boxed{1}$	$\boxed{2}$	$\boxed{3}$	$\boxed{4}$	$\boxed{5}$	Rot(6,7,6)	$\boxed{8}$	
$t=14$	$\boxed{1}$	$\boxed{2}$	$\boxed{3}$	$\boxed{4}$	$\boxed{5}$	$\boxed{6}$	Rot(7,8,7)	
$t=15$	$\boxed{1}$	$\boxed{2}$	$\boxed{3}$	$\boxed{4}$	$\boxed{5}$	$\boxed{6}$	$\boxed{7}$	8

Au top $2n-1$, le processeur P_k contient la ligne k. Il faudrait éventuellement vider le réseau.

▷ Question 2, page 1

On voit bien que le réseau précédent peut être replié, pour obtenir un réseau bidirectionnel :

$$\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{5}\boxed{6}\boxed{7}\boxed{8} \rightleftarrows P_1 \rightleftarrows P_2 \rightleftarrows P_3 \rightleftarrows P_4 \,.$$

Au top n, on renverse le sens de circulation, pour obtenir sur l'exemple avec $n = 8$:

Top	P_1	P_2	P_3	P_4
$t=1$	$\boxed{8}$			
$t=2$	Rot(7,8,1)			
$t=3$	Rot(6,7,1)	$\boxed{8}$		
$t=4$	Rot(5,6,1)	Rot(7,8,2)		
$t=5$	Rot(4,5,1)	Rot(6,7,2)	$\boxed{8}$	
$t=6$	Rot(3,4,1)	Rot(5,6,2)	Rot(7,8,3)	
$t=7$	Rot(2,3,1)	Rot(4,5,2)	Rot(6,7,3)	$\boxed{8}$
$t=8$	Rot(1,2,1)	Rot(3,4,2)	Rot(5,6,3)	Rot(7,8,4)
$t=9$	Rot(2,3,2)	Rot(4,5,3)	Rot(6,7,4)	$\boxed{8}$
$t=10$	Rot(3,4,3)	Rot(5,6,4)	Rot(7,8,5)	
$t=11$	Rot(4,5,4)	Rot(6,7,5)	$\boxed{8}$	
$t=12$	Rot(5,6,5)	Rot(7,8,6)		
$t=13$	Rot(6,7,6)	$\boxed{8}$		
$t=14$	Rot(7,8,7)			
$t=15$	$\boxed{8}$			

La première ligne ressort du réseau par la gauche au top $t = n+1$, et la dernière ligne ressort au top $2n$.

▷ Question 3, page 1

Les deux architectures sont identiques ! En fait, le plongement de la grille torique dans l'hypercube à l'aide des codes de Gray est un isomorphisme complet, puisque chaque processeur a exactement six voisins. Dans le même ordre d'idées, on pourra remarquer qu'un hypercube de dimension 4 et un tore 2D de taille 4×4 sont identiques.

▷ Question 4, page 1

La réponse est négative dès que n est grand, et ceci pour toute valeur de r, même exponentielle en n. Pour le voir, remarquons que l'arbre binaire complet à $2^n - 1$ sommets est de hauteur $n-1$: tous les sommets de l'arbre sont à une distance au plus $n-1$ de la racine.

Par contre, dans un grille torique de taille $r \times r$, avec r arbitrairement grand, il y a, en majorant largement, au plus $4n^2$ sommets situés à une distance au plus n d'un sommet donné S : en effet, ceux-ci sont nécessairement à l'intérieur d'un carré centré en S et de côté $2n$.

Il suffit de choisir pour S l'image de la racine de l'arbre dans un plongement éventuel pour aboutir à une contradiction dès que $2^n - 1 \geqslant 4n^2$, c'est-à-dire pour $n \geqslant 9$ (on pourrait affiner cette borne, mais c'est sans importance).

▷ Question 5, page 1

Le nombre de processeurs est clairement égal à $p = m \cdot 2^m$. Un sommet peut se représenter par le couple $s =< c, i >$ où c est la composante hypercube de s, à savoir une suite de m bits (qui s'interprète comme le numéro du cycle dans l'hypercube) et $i \in [\![0, m-1]\!]$ est la composante cycle de s, c'est-à-dire la position du sommet dans le cycle. Les sommets $< c, i >$ et $< c', i' >$ sont liés si et seulement si :

- $w = w'$ et $i - i' \equiv \pm 1[m]$ (arête de cycle) ou
- $i = i'$ et w et w' diffèrent sur le i-ème bit (arête d'hypercube).

Pour majorer le diamètre, cherchons à majorer la distance entre deux sommets arbitraires. On peut d'abord se ramener à chercher un chemin du sommet $(0,0)$ à un sommet donné (w,i) : comme sur l'hypercube, on peut effectuer un XOR bit à bit des composantes hypercube des sommets.

Pour aller de $(0,0)$ à (w,i), on va, tout comme sur l'hypercube, corriger les bits de w, par exemple à partir des poids faibles : pour chaque 1 dans l'écriture de w, on traversera l'arête hypercube correspondante. Bien sûr, il faudra avancer d'un cran, ou de plusieurs, sur le cycle avant de traverser une nouvelle arête hypercube. Pour atteindre le cycle numéroté w, on obtient une majoration en $2m-1$ arêtes : au plus m arêtes hypercubes à corriger, et $m-1$ crans de cycle, un pour chaque arête sauf la première. Reste alors à rejoindre la position i sur le cycle : mais le diamètre d'un anneau de taille m est $\lfloor \frac{m}{2} \rfloor$, d'où la majoration du diamètre $D \leqslant 2m - 1 + \lfloor \frac{m}{2} \rfloor$.

Notons que chaque sommet est de degré trois, mais que le diamètre reste logarithmique en le nombre total de processeurs (on a $m \leqslant \log p$).

▷ Question 6, page 1

Nous n'étions pas loin avec la majoration précédente ! Pour $m = 3$, il suffit d'inspecter la figure 1 pour vérifier que le diamètre est bien 6.

Pour $m > 3$, la preuve peut être formulée de la façon suivante. Pour aller de $(0,0)$ à (w,i), il va falloir traverser au moins $|w|$ arêtes d'hypercube, où $|w|$ est le nombre de 1 dans l'écriture de w. Admettons pour l'instant (c'est intuitif) qu'un plus court chemin empruntera exactement w arêtes d'hypercube. Représentons le cycle numéro 0 en entourant les sommets qui correspondent aux position des 1 dans l'écriture de w. Par exemple si $m = 8$ et $w = 11000110$, on entoure les sommets en position 1, 2, 6 et 7 :

$$\begin{array}{ccccccc} 0 & \rightarrow & \boxed{1} & \rightarrow & \boxed{2} & \rightarrow & 3 \\ \uparrow & & & & & & \downarrow \\ \boxed{7} & \leftarrow & \boxed{6} & \leftarrow & 5 & \leftarrow & 4 \end{array}$$

L'idée sous-jacente est de traverser l'arête d'hypercube la première fois qu'on arrive sur un sommet entouré. La contrainte est alors de trouver un chemin de longueur minimale sur le cycle qui parte de 0 et qui se termine en i, en passant par tous les sommets entourés au moins une fois. La longueur totale du routage de $(0,0)$ à (w,i) sera égale à $|w|$ plus la longueur du chemin sur le cycle. Sur l'exemple si $i = 5$, un plus court chemin sur le cycle est :

$$0 \rightarrow 1 \rightarrow 2 \rightarrow 1 \rightarrow 0 \rightarrow 7 \rightarrow 6 \rightarrow 5,$$

de longueur 7. Il faut encore ajouter $|w| = 4$ pour obtenir la longueur du routage.

Avec cette représentation, on voit bien qu'il est inutile d'emprunter des arêtes hypercube en nombre supplémentaire. Le pire des cas est obtenu pour $w = 2^m - 1$ (tous les bits sont à 1, donc tous les sommets sont entourés) et $i = \lfloor \frac{m}{2} \rfloor$: le plus court chemin issu de 0 qui visite tous les sommets et termine en i est

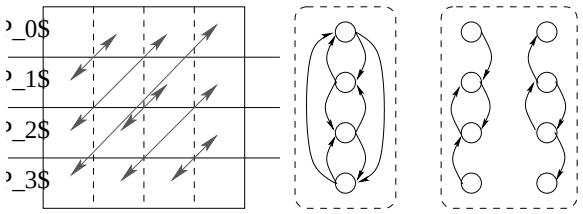

FIG. 2 – Mouvements de données sur une ligne de 4 processeurs.

de longueur $m + \lfloor \frac{m}{2} \rfloor - 2$, quantité à laquelle il faut ajouter m pour les arêtes d'hypercube. Sur l'exemple avec $m = 8$, un plus court chemin de 0 à 4 est :

$$0 \to 7 \to 6 \to 5 \to 6 \to 7 \to 0 \to 1 \to 2 \to 3 \to 4,$$

de longueur $10 = 8 + 4 - 2$.

▷ Question 7, page 2

Regardons d'abord quelles communications sont nécessaires à la transposition de la matrice sur une ligne de 4 processeurs (voir figure 2).

Supposons p divisible par n. Chaque processeur doit communiquer une donnée particulière (un bloc de taille $\frac{n}{p} \times \frac{n}{p}$ à tous les autres processeurs. C'est un échange total personnalisé. En regroupant les communications en fonction de la distance à laquelle se trouvent les processeurs impliqués, on obtient les phases figure 3.

On notera que dans chacune de ces phases (à part peut-être la dernière) chacun des processeurs échange des données avec deux autres processeurs et qu'ainsi, comme la distances entre chaque couple de processeurs est constante, en « pipelinant » les communications, chaque lien de communication est utilisé en permanence et chaque message emprunte le chemin le plus court possible.

La quantité de communications nécessaires est donc égale à :

$$\sum_{k=1}^{\lfloor \frac{p}{2} \rfloor} k(\beta + \tau \frac{n^2}{p^2}) = \frac{1}{2} \left(\left\lfloor \frac{p}{2} \right\rfloor \left(\left\lfloor \frac{p}{2} \right\rfloor + 1 \right) \right) \left(\tau \frac{n^2}{p^2} + \beta \right) \sim \tau \frac{n^2}{8} + \beta \frac{p^2}{8} .$$

▷ Question 8, page 2

De même que précédemment, regroupons les communications en fonction de la distance séparant les processeurs (voir figure 4).

En décomposant les communications le long des arcs, on peut s'apercevoir qu'à une étape donnée, un processeur participe à une seule des communications. Du coup, le temps des communications nécessaires à la mise en œuvre d'une transposition est égal à :

$$2 \left\lfloor \frac{q}{2} \right\rfloor \cdot \left(\tau \frac{n^2}{q^2} + \beta \right) \sim \tau \frac{n^2}{2\sqrt{p}} + \beta \frac{\sqrt{p}}{2}$$

Cet algorithme est clairement optimal puisque son temps d'exécution est aussi égal au temps de communication d'une portion de matrice entre les deux processeurs les plus éloignés.

▷ Question 9, page 2

La structure récursive des hypercubes permet de mettre facilement en œuvre la transposition récursive illustrée figure 5. Du coup, le temps des communications nécessaires à la mise en œuvre d'une transposition est égal à :

$$k \cdot \left(\tau \frac{n^2}{p} + \beta \right) \sim \tau \frac{(\log p) n^2}{p} + \beta \log p$$

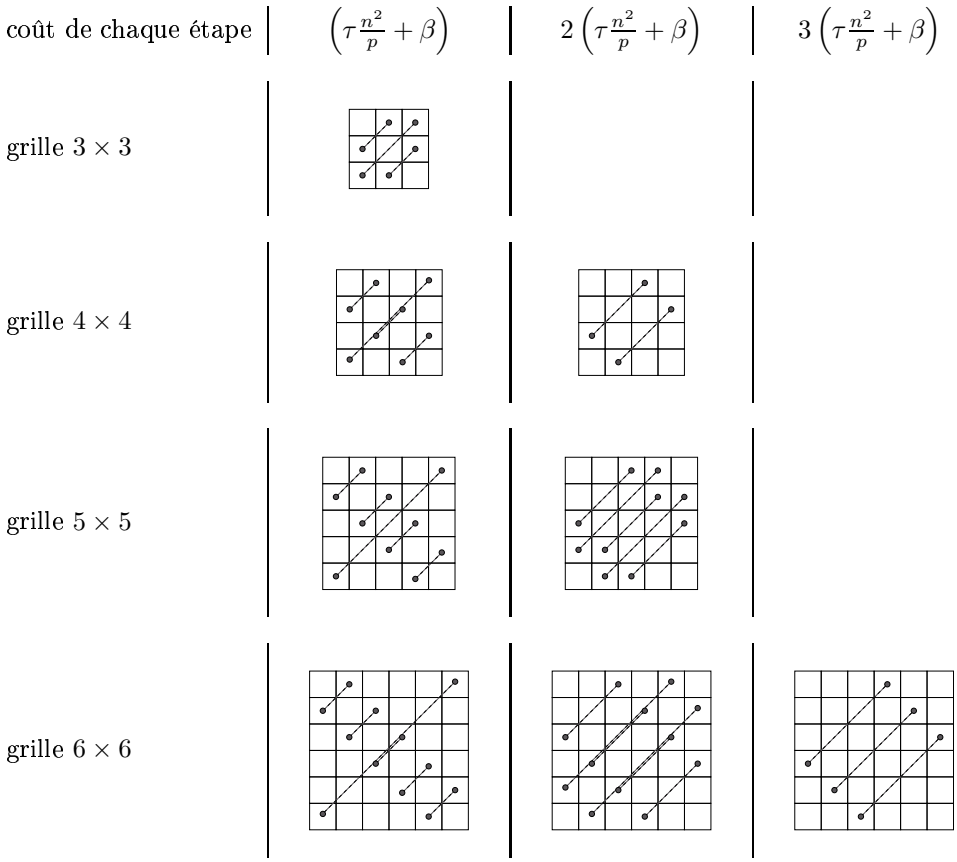

FIG. 3 – Organisation des communications sur des lignes de processeurs.

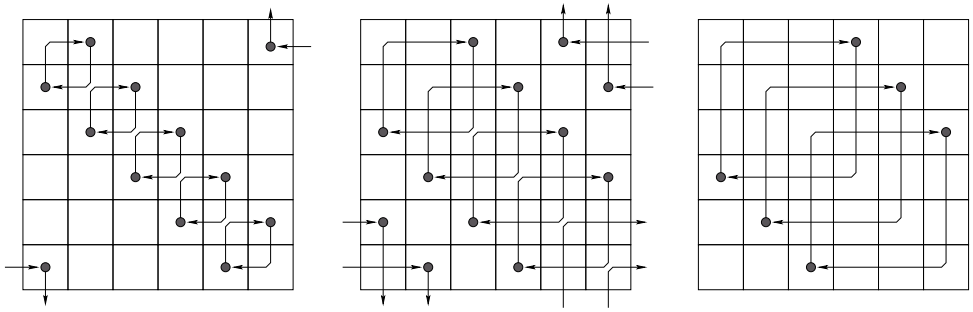

FIG. 4 – Mouvements de données sur une grille de 6×6 processeurs.

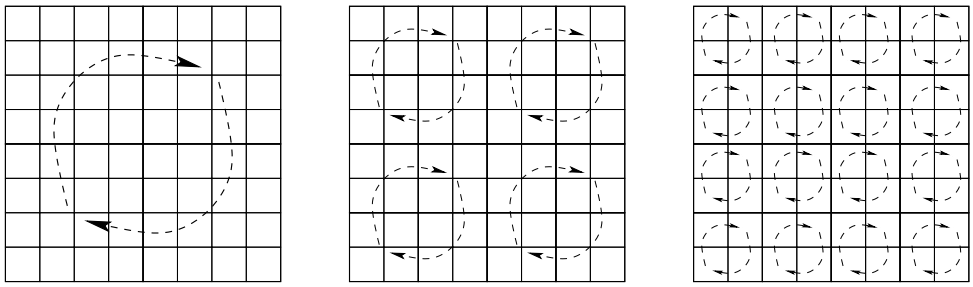

FIG. 5 – Mouvements de données sur un hypercube de dimension 6.