Partiel MIM2 – Algorithmes et architectures parallèles

Yves Robert

Lundi 29 Octobre 2001, 10h15–12h15

Au menu des P-RAM, des réseaux d'interconnexion et de l'algorithmique parallèle. Choisissez ! La seule partie vraiment difficile est indiquée comme telle.

1 Procédure mystère sur P-RAM

1.1 Procédure SPLIT

On définit les deux opérateur suivants pour un tableau $A = [a_0, a_1, \ldots, a_{n-1}]$ de n entiers :
PRESCAN(A) renvoie le tableau $[0, a_0, a_0 + a_1, a_0 + a_1 + a_2, \ldots, a_0 + a_1 + \ldots + a_{n-2}]$
SCAN(A) renvoie le tableau $[a_0, a_0 + a_1, a_0 + a_1 + a_2, \ldots, a_0 + a_1 + \ldots + a_{n-1}]$

1. Rappeler brièvement comment réaliser ces deux opérateurs sur une P-RAM EREW
2. Etant donné un tableau de booléens FLAGS, dire ce que fait la procédure :

```
procedure split(A, Flags)
  Idown ← prescan(not(Flags))
  Iup ← n - reverse(scan(reverse(Flags)))
  for each index i in // do
      if Flags(i) then Index[i] ← Iup[i] else Index[i] ← Idown[i]
  result ← permute(A,Index)
```

Tous les noms sont intuitifs. En particulier, REVERSE renverse le tableau. L'horrible expression REVERSE(SCAN(REVERSE(FLAGS))) fait simplement un SCAN à partir de la fin du tableau FLAGS, dont les éléments sont considérés comme des entiers. Voici un exemple :

$$\begin{array}{lcl}
A & = & [\ 5\ \ 7\ \ 3\ \ 1\ \ 4\ \ 2\ \ 7\ \ 2\ ] \\
Flags & = & [\ 1\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ ] \\
Idown & = & [\ 0\ \ 0\ \ 0\ \ 0\ \ \boxed{0}\ \ \boxed{1}\ \ 2\ \ \boxed{2}\ ] \\
Iup & = & [\ \boxed{3}\ \ \boxed{4}\ \ \boxed{5}\ \ \boxed{6}\ \ 7\ \ 7\ \ \boxed{7}\ \ 8\ ] \\
Index & = & [\ 3\ \ 4\ \ 5\ \ 6\ \ 0\ \ 1\ \ 7\ \ 2\ ] \\
result & = & [\ 4\ \ 2\ \ 2\ \ 5\ \ 7\ \ 3\ \ 1\ \ 7\ ]
\end{array}$$

Quelle est la complexité de la procédure SPLIT ?

1.2 Procédure mystère

On considère la procédure suivante :

```
procedure myst(A, number-of-bits)
  for i = 0 to NumberOfBits -1
     A(i) ← tableau des bits en position i des éléments de A
     A ← split(A,A(i))
```

1. Faire tourner la procédure sur $A = [5, 7, 3, 1, 4, 2, 7, 2]$ avec NumberOfBits=3. Au début : $A(0) = [1, 1, 1, 1, 0, 0, 1, 0]$ ...
2. Que fait la procédure mystère ?
3. Avec des entrées de taille $O(\log n)$ bits, quelle est la complexité avec n processeurs ? et avec seulement p processeurs ? quelles sont les valeurs de p les plus intéressantes ?

2 Communication dans les réseaux

2.1 Butterfly

Un réseau "butterfly" de dimension r est un réseau composé de $(r+1)2^r$ noeuds organisés en 2^r lignes de $r+1$ niveaux. Un noeud est désigné par une paire (w, i) où w est un entier codé en représentation binaire sur r bits qui numérote la ligne du noeud, et où i numérote le niveau du noeud ($0 \leq i \leq r$). Deux noeuds (w, i) et (w', i') sont reliés par un arc si et seulement si $i' = i + 1$ et l'une des deux conditions suivantes est remplie :

- $w = w'$
- w et w' ne diffèrent que par le i-ème bit.

Sur la figure 1 est représenté le réseau butterfly de dimension 3.

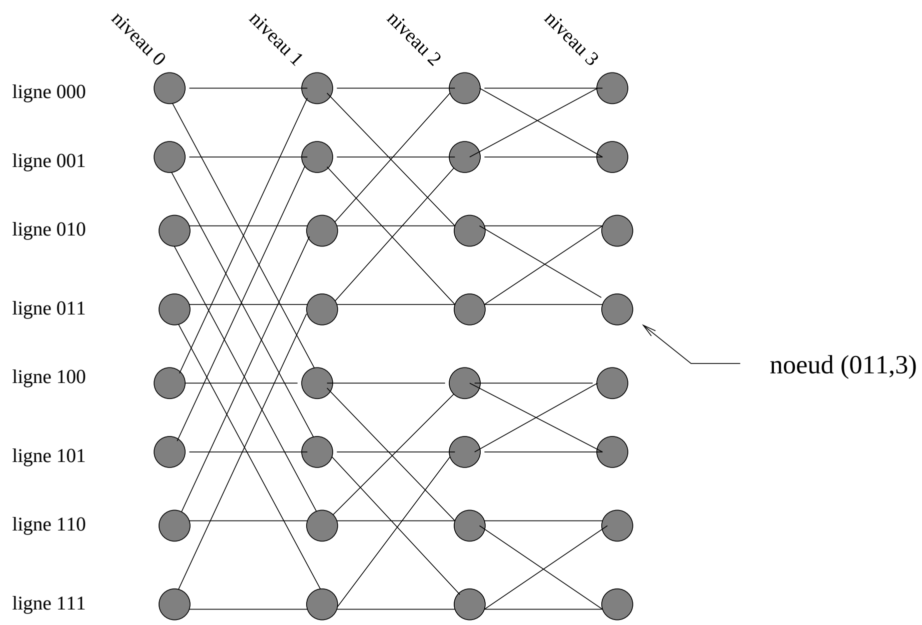

FIG. 1 – Le réseau butterfly de dimension 3.

1. Quelle est l'architecture du réseau obtenu lorsque l'on regroupe les noeuds d'une même ligne en un seul noeud (et que l'on enlève les arcs en trop qui sont dupliqués) ?
2. Le butterfly a une structure récursive. Donner deux moyens d'obtenir deux butterfly de dimension $r-1$ à partir d'un réseau butterfly de dimension r (on ne cherchera pas à prouver formellement la décomposition).
3. Donner un algorithme de routage pour amener un message d'un noeud $(w, 0)$ du niveau 0 au noeud (w', r) du niveau r.

4. Écrire une procédure SPMD `route(source,dest,msg)` exécutée par chaque noeud, qui effectue le routage d'un message `msg` du noeud `source` au noeud `dest`. On supposera que chaque noeud communique avec ses voisins à l'aide des fonctions `send(dest, msg)` et `receive(source, msg)` et connaît son adresse (`my_id`). Les adresses `dest, source, my_id` sont des paires (w, i).

2.2 Benes (difficile)

Un réseau de Benes de dimension r est composé de deux réseaux butterfly mis "dos à dos". Les noeuds des niveaux r de chaque butterfly sont fusionnés : chaque niveau comprend alors $2r + 1$ noeuds. sur un même niveau. La figure 2 montre un réseau de Benes de dimension 3. On supposera ici que les noeuds du réseau sont uniquement des "switches 2×2", c'est à dire qu'ils peuvent se configurer en croix ou en lignes parallèles pour transmettre les données.

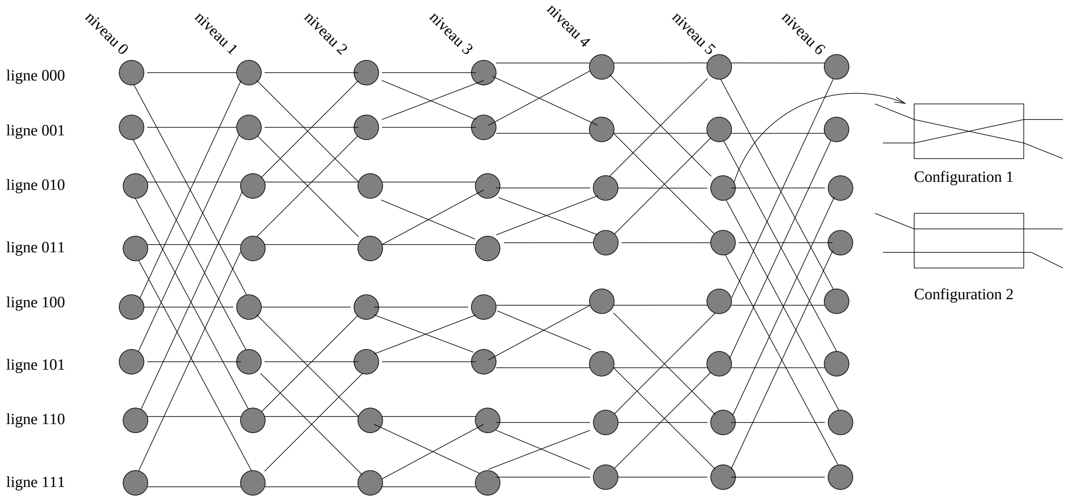

FIG. 2 – Le réseau de Benes de dimension 3.

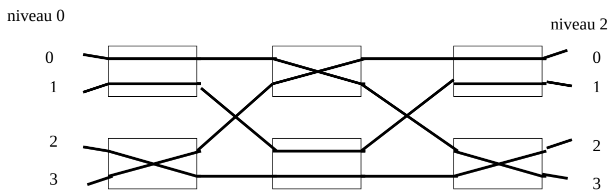

FIG. 3 – Configuration du réseau de Benes de dimension 1.

Montrer (par récurrence sur la dimension r du réseau) que le réseau de Benes peut être configuré pour réaliser une permutation arbitraire : étant donnée une permutation quelconque π des 2^{r+1} premiers entiers (de 0 à $2^{r+1} - 1$), il existe une configuration des switches qui permet de connecter simultanément l'entrée i du réseau à sa sortie $\pi(i)$. Par exemple, sur la figure 3 est représentée une configuration des switches d'un réseau de Benes de dimension 1 réalisant la permutation $\pi = (0, 1, 2, 3) \to (3, 1, 2, 0)$.

3 Algorithmique parallèle

On veut concevoir un algorithme parallèle pour la résolution d'un système triangulaire sur un anneau de processeurs.

Notations : Soit $A = (a_{ij})_{0 \leq i,j \leq n-1}$ une matrice triangulaire inférieure d'ordre n et b un vecteur à n composantes. On veut résoudre le système $Ax = b$. La machine-cible est un anneau orienté de p processeurs P_q, $0 \leq q < p$.

Hypothèses : On suppose la matrice stockée de manière distribuée dans les processeurs. Plus précisément, on distribue les lignes de la matrice aux processeurs, et on fait les hypothèses suivantes :
- n est divisible par p
- Le temps pour communiquer L mots entre 2 processeurs voisins est $\beta + L\tau_c$
- Le temps pour une multiplication suivie d'une addition ou pour une soustraction suivie d'une division est τ_a
- On peut calculer et communiquer des données en parallèle.

Questions : Concevoir un algorithme parallèle :

1. Fonction d'allocation : quelles lignes de la matrice sont allouées à chaque professeur ? Que supposez-vous pour le vecteur b ?
2. Programme des processeurs
3. Etude des performances.

Votre algorithme est-il optimal ? Si oui, le démontrer, si non, essayer de l'améliorer ! Ne pas hésiter à proposer plusieurs variantes.

Bonus : Peut-on faire mieux sur un tore 2D que sur l'anneau ?