Contents

1 Réseaux d'interconnections dynamiques

- connexion par bus
- crossbar
- réseaux multi-étages

1.1 Règle de connexion

C'est un shift circulaire sur la gauche (shuffle inverse)

- Les étages sont reliés par la permutation suffle inverse (ou shift circulaire à gauche)
- Les sorties des commutateurs ont une propriété: les sorties du haut sont un nombre pair et celles du bas sont un nombre impair. En binaire, ces nombres se terminent par un 0 ou un 1, respectuvement.

2 Algorithme de routage

Soit $o_1, o_2 \dots o_n$ les n bits de l'entrée (origine du message) et $d_1, d_2, \dots, d_n$ l'adresse de destination.

$o_1, o_2, \dots o_n \rightarrow o_2, o_3, \dots o_n, o_1$ (lien entre entrée et commutateur) Si on sort par le haut: on aura $o_2, o_3, \dots, o_n 0$ Si on sort par le bas: on aura $o_2, o_3, \dots, o_n 1$ Ce qui nous donne $o_2, o_3, \dots, o_n, d_1$

2ème étage: -> $o_3, o_4, \dots, o_n, d_1, o_2$ -> $o_3, o_4, \dots, o_n, d_1, d_2$ ... -> $d_1, d_2, ..., d_n$ ce qui est bien l'adresse de destination

Combien faut-il d'étages si on a N entrées et N sorties ?

$$n = log_2(N)$$

Chaque étage contient combien de switchs 2x2 ?

Il en faut $\frac{N}{2}$

Coût de ce réseau: $O(\frac{N}{2} \times log_2 N = O(Nlog_2 N)$ (Même chose que hypercube)

Les réseaux multi-étages sont moins chers que le crossbar $O(N^2)$, mais il est aussi moins performant: on dit que c'est un réseau blocant: un choix de chemin input->output peut en empêcher un autre de se réaliser...

On peut facilement modifier les commutateurs de 2 pour faire un broadcast

Le broadcast se fait alors avec 1 traversée du réseau, soit donc en $log_2 N$ étapes.

On remarque aussi qu'on peut envoyer des messages en pipeline sans attendre qu'un message soit arrivé pour envoyer le suivant.

2.1 Les réseaux Fat Tree

C'est une forme de réseau multiétage où des commutateurs sont reliées selon une structure d'arbre. La particularité est que la bande passante augmente à mesure qu'on s'approche de la racine.

2.2 Routage sur un arbre

Noeud X: sous-noeud 2X gauche... Sous-noeud 2X + 1 droite....

Supposons que l'on veut connaître l'ancêtre commun. C'est le préfixe commun (ex: 5: 101 et 8: 1000 préfixe commun: 10 -> ancêtre commun de 5 et de 8: 2

Plusieurs chemins permet de rendre le réseau moins blocant.

2.3 Définition

Puissance de calcul d'un processeur: R = Flop/s ou op/s ou cycle ou fréquence

Travail: W = nombre d'opérations réalisées pour résoudre un problème

Il y a le lien évident entre W et R:

W = R * T où T est le temps d'exécution.

Degré de parallélisme: p(t): c'est le nombre de PE actif au moment t d'une exécution parallèle. (shéma du prof)

En parallèle, le temps d'exécution est donné par: $T_{par} = T_{max} - T_{min}$

$$W = \int_{T_{min}}^{T_{max}} p(t) R dt$$

En séquentiel, le temps nécessaire serait

$$T_{seq} = \frac{W}{R} = \int_{T_{min}}^{T_{max}} p(t) dt$$

On définit alors le speedup S comme:

$$S = \frac{T_{seq}}{T_{par}}$$

Combien de fois va-t-on plus vite en parallèle ?

Ici, on a $T_{seq} = \int_{T_{min}}^{T_{max}} p(t)dt$ et $T_{par} = T_{max} - T_{min}$

$$S = \frac{1}{T_{max} - T_{min}} \int_{T_{min}}^{T_{max}} p(t)dt$$

Le speed up est donc le degré de parallélisme moyen

Overhead: PE réservés, mais inactifs

L'overhead est une mesure du travail perdu en raison des processeurs unactifs durant l'exécution parallèle. On exprime l'overhead Δ comme

$$\Delta = W_{par} - W_{seq}$$

$$= (p_{max}T_{par} - T_{seq})R$$

Quelles sont les sources d'overhead en parallélisme ?

- communication/coordination (overhead peut prendre plus de temps que le calcul lui-même...
- mauvais équilibrage de charge, synchronisation
- Algorithme: certains algorithmes séquentiels se parallélisent mal et on en prend un autre en parallèle. Ex: quick-sort ($O(nlogn)$) (difficile à paralléliser vs bubble sort ($O(n^2)$) (facile à paralléliser)

Revisitions la définition du speedup:

$$S = \frac{T_{seq}}{T_{par}}$$

où T_{seq} est le temps du meilleur algorithme connu pour le problème.

$$T_{seq} \neq T_{par}(p = 1)$$

Cela signifie que le temps séquentiel est différent du temps parallèle exécuté avec un seul processeur.

Mais cette définition formelle ne s'applique pas aux grosses machines HPC qui tournent des codes trop grands pour être exécutable en séquentiel

$$S = \frac{T_{par}(p = p_{ref})}{T_{par}(p)}$$

2.4 Définition

Efficacité: $E = \frac{S}{p}$ (p: nombre de processeurs...).

On se compare à un speedup idéal qui serait $S = p$...

En général, $S < p$ car il y a de l'overhead...

2.5 Relations utiles

1. $$E = \frac{T_{seq}}{pT_{par}} \Rightarrow T_{par} = \frac{T_{seq}}{E \times p}$$

   C'est comme si on avait un nombre effectif de processeurs $p' = Ep$

   Aussi, en multipliant par R

   $$T_{par} = \frac{T_{seq}R}{EpR} = \frac{W_{seq}}{pER}$$

   C'est comme si en parallèle les processeurs avaient une puissance réduite $R' = ER$

2. $$W_{par} = pRT_{par} = W_{seq} + \Delta$$

   avec Δ l'overhead.

   D'où:

   $$T_{par} = \frac{W_{seq}}{Rp} + \frac{\Delta}{Rp}$$

   et ainsi:

   $$S = \frac{T_{seq}}{T_{par}} = \frac{W_{seq}/R}{\frac{W_{seq}}{Rp} + \frac{\Delta}{Rp}} = \frac{p}{1 + \frac{\Delta}{W_{seq}}}$$

Donc le speedup s'éloigne du speedup idéal $S = p$ en raison de l'importance de l'overhead en regard du travail qu'on parallélise.

2.6 Loi d'Amdahl (strong scaling)

Cette loi donne une vision pessimiste du potentiel de la parallélisation

On va faire l'hypothèse que le code se parallélise idéalement sur une partie et pas du tout sur une autre partie. On va avoir:

- αW fraction du travail non parallélisable
- $(1 - \alpha)W$ travail qui se parallélise idéalement.

$$T_{par} = \frac{\alpha W}{R}(temps\ séquentiel) + \frac{(1-\alpha)W}{pR}(temps\ parallèle)$$

$$T_{seq} = \frac{W}{R}$$

$$S = \frac{T_{seq}}{T_{par}} = \frac{\frac{W}{R}}{\alpha\frac{W}{R} + \frac{(1-\alpha)W}{pR}} = \frac{1}{\alpha + \frac{1-\alpha}{p}} \leq \frac{1}{\alpha}$$

Si $\alpha = 10\%$ et $p = 1000$ sur les 90% restant, on a que

$$S = 9.91 \leq 10$$

et

$$E = \frac{S}{p} = 10^{-2}$$