Contents

1 Réseaux d'interconnexion

Modèle de performance des communications "send-receive"

Un message traverse le réseau entre la source et la destination. Combien de temps cela prend ?

On va voir deux techniques de commutation

- store-and-forward: mauvaise performance
- cut-though: bonne performance

1.1 Store-and-forward

On représente notre système comme une suite de noeuds à traverser entre la source et la destination.

On a le temps de transfert entre 2 noeuds de $t + \frac{L(bytes)}{W(bandepassante)}$

On a également un temps de startup ou de latence: t_s.

Propagation du message: $T_{sf} = t_s + \frac{nL}{W}$ où n est le nombre de noeuds à traverser

1.2 Cut-through

On fait passer 1 byte à la fois... on a donc t + 1/W entre 2 noeuds

On a: $T_{cut-through} = t_s + \frac{n \text{ (temps du premier byte)} + L - 1}{W}$

Dans le cas où n est petit (c'est borné par le diamètre) par rapport à L (cas fréquent car en général L est grand...).

On peut alors approximer

$$T_{ct} = t_s + \frac{L}{W}$$

Donc le temps de communication ne dépend plus de l'origine et de la destination... Et si $\frac{L}{W} >> ts(\approx 1\mu s)$, alors

$$T_{ct} = \frac{L}{W}$$

Il vaut mieux envoyer peu de grands messages que beaucoup de petits messages

Bande baobab 2Gb/s

latence 2.4 micro secondes

Tous les processeurs ne sont pas reliés fortement entre eux dans baobab...

1.3 3.3 Réseaux statiques

C'est une topologie possible pour un réseau d'interconnexions: c'est un graphe dont les sommets (ou noeud) sont les PE + routeur et les arcs sont les liens physiques reliant les processeurs.

On va considérer plusieurs topologies statiques et voir leurs performances à travers les propriétés vues précédement.

1.3.1 Anneau de processeurs

- N: nombre de processeurs
- Degré du graphe: 2 (c'est indépendant du nombre de processeurs...) Ici on n'est pas content
- Diamètre: N/2 (augmente linéairement avec N) Ici on n'est pas content
- Largeur bisectionnelle: 2 (indépendant du nombre de processeurs...) Ici on n'est pas content
- Prix: quelle est la dépendance avec N ? O(N) Ici on est content
- Scalabilité: du point de vue architectural, on peut faire des anneaux aussi grands qu'on veut... Donc oui, c'est scalable... Ici on est content (Mais du point de vue performance, cela n'ira pas...)

Graphe connexe:

- degré: O(N) Content
- diamètre: 1 Content
- largeur bisectionnelle: O(N) Content
- prix: $O(N^2)$ (le nb de liens augmente en $O(N^2)$... Pas content
- scalabilité: Non car le nb de liens augmente et le matériel change avec la taille pas content

1.3.2 Grille de processeurs 3D

C'est une grille périodique ou tore...

- degré: 4
- diamètre: O(sqrt(N)) cela croît plus lent que linéaire
- largeur bisectionnelle: O(sqrt(N))
- prix: O(N)

- scalabilité: oui

1.3.3 Grille de processeur en D-dimensions

- N: nb processeurs
- degré: 2D
- diamètre:$DN^{1/d}$ Plus D est grand, plus le diamètre augmente lentement avec N.
- largeur bisectionnelle: En coupant en 2 le réseau, on crée une face à (D - 1) dimensions avec $N^{1/d}$ noeud par côté. Elle contient $(N^{1/D})^{d-1}$ processeurs d'où part une connexion vers l'autre moitié.

  $\Rightarrow N^{(d-1)/d} \to N$ si d est grand
- prix: O(DN)
- scalabilité: oui sauf s'il faut de plus en plus de fils très long...

Exemple: IBM BlueGeve avait une dimension de 5.

1.3.3.1 Routage et broadcast sur une grille 2D

L'algorithme de routage consiste à bouger horizontalement du bon nombre de noeuds, et aussi verticalement, dans un ordre arbitraire: il y a plusieurs chemins optimaux. Ce qui offre des alternatives en cas de nombreux messages qui emprunte le même noeud intermédiaire (Routage facile... et locale (à chaque position, on peut déterminer du chemin à suivre...))

1.3.3.2 Broadcast

- single port: On diffuse le message horizontalement, puis verticalement

  Il faut faire un broadcast sur les noeuds horizontaux d'abord, puis verticaux (O(sqrt(N)))
- multiport: On obtient le même résultat: O(sqrt(N)), mais la complexité pour déterminer à quel autre noeud on envoie le message est plus grande...

1.3.4 HyperCube

C'est un peu une topologie statique de référence au vue de ses performances

1.3.4.1 Construction d'un hypercube en fonction de sa dimension d.

- 1 seul noeud: hypercube de dimension d = 0.
- 2: dimension d = 1
- 4: dimension d = 2
- 8: dimension d = 3

On construit un hypercube de dimension d avec 2 hypercubes de dimension d - 1 dont on relie les noeuds correspondants

Combien y a-t-il de noeuds dans un hypercube de dimension d ?

$$N = 2^d \Rightarrow d = log_2(N)$$

On peut voir un hypercube comme un cube de dimension d mais d'arrête 1.

Pour augrmenter la taille de l'hypercube (N = nb de processeurs), il faut augmenter sa dimension d. Cela contraste avec la grille où on augmente N en augmentant la taille de l'arrête.

- degré: d
- diamètre: d également: on parcours chacune des d arrêtes de taille 1.
- largeur bisectionnelle: $\frac{N}{2} = 2^{d-1} = O(N)$
- prix: $O(dN) = O(log_2(N) \times N) \rightarrow$ croissance plus rapide que linéaire... => difficile de faire des hypercubes de taille trop grande...
- scalabilité: non car le hardware change en fonction de la taille de l'hypercube: il faut des routeurs avec plus de ports ...

Exemple: Connection Machine (1990) -> hypercube de dimension 16 (à Genève on a eu une machine hypercube avec une dimension 13)

1.3.4.2 Numérotation des noeuds

1 des 2 sous-hypercubes va avoir en plus un préfixe 1, et l'autre un préfixe 0...

Avec cette construction, 2 noeuds voisins ont leur numérotation qui ne diffère que de 1 bit. C'est le bit qui est dans la dimension dans laquelle les noeuds sont reliés qui diffère...

Il y a beaucoup d'algorithmes parallèles qui supposent que les processeurs sont en topologie d'hypercube. Cette topologie est donc très utilisée pour organiser les processeurs d'une manière algorithmique...