Chapitre 1

Introduction

Pourquoi faire du multicoeurs :
- Limite thermique
- Limite des accès mémoires
- Limite de l'intégration (mettre $\approx 10^{12}$ mots sur une surface de $0.3^2 mm^2$)

La solution est le CMP, une architecture MIMD (Multiple Instruction Stream, Multiple Data stream), il s'agit d'un ensemble de thread qui exécute un graphe de précédence.

1.1 Loi d'Amdhal

Le gain de vitesse (speed-up) pour une architecture à n coeurs est donné par $S(n) = \frac{T_1}{T_n}$ où T_n est le temps pour exécuter le problème avec n coeurs. Idéalement, il faudrait avoir $S(n) = n$: aller n fois plus vite. Mais c'est rarement le cas !

Tout programme contient une partie séquentielle *seq* et une partie *par* pouvant être exécutée en parallèle. Soit p la partie séquentielle du problème : $p = \frac{seq}{seq+par}$.

$$S(n) = \frac{T_1}{T_1 p + \frac{(1-p)T_1}{n}} = \frac{1}{p + \frac{1-p}{n}}$$

Exemple : 60% concurrent et 40% séquentielle : $S(10) = \frac{1}{0.4+\frac{0.6}{10}} = 2.17$, $S(10) = \frac{1}{0.99+\frac{0.01}{10}} = 9.17$.

Conclusion : le petit % séquentielle influence fortement le gain en vitesse fourni par un CMP.

1.2 Efficacité

L'efficacité $E(n)$ est l'utilisation moyenne des n coeurs pour exécuter l'application : $E(n) = \frac{S(n)}{n}$. $E(n) = 1$ implique que $S(n) = n$. Généralement une augmentation de $S(n)$ (en donnant des coeurs à l'application) se réalise au détriment de l'efficacité.

1.3 Caractérisation du parallélisme

— p : fraction séquentiel de l'application
— m_{min}
— m_{max}
— p_i proportion de l'exécution avec i coeurs
— A : paralélisme moyen de l'application

$$A = \sum_{i=m_{min}}^{m_{max}} ip_i$$

A peut se définir comme étant
— égal au nombre moyen de coeurs utilisés durant l'exécution de l'application si $n \geq m_{max}$.
— égal au gain de vitesse $S(\infty) = \frac{T_1}{T_\infty} = \frac{AT_\infty}{T_\infty}$.
— le rapport entre la somme du temps d'exécution de toutes les actions du graphe de précédence de l'application et le chemin le plus long de la racine aux feuilles de ce graphe.

1.4 Relation entre $S(n)$ et A

Soit A le parallélisme moyen d'une application, $S(n)$ son gain de vitesse pour n coeurs et $E(n)$ l'efficacité des n coeurs. Alors $S(n) \geq \frac{nA}{n+A-1}$ et $E(n) \geq \frac{A}{n+A-1}$.

La programmation concurrente multicoeurs est difficile :
— Limitation de la loi d'Amdhal.
— Idéalement il faut maintenir une efficacité élevée.
— Le dimensionnement de la granularité de calcul est difficile (granularité = exécution fait entre 2 points de synchronisation).
— la localité des données freine la vitesse (conserver le maximum de données en antémémoire).
— le balancement de la charge de chaque thread n'est pas évident.
— le partage de données et la synchronisation entre les threads doivent être rapide.

Chapitre 2

Exclusion mutuelle

Un thread est une suite ordonnée d'événemments.

2.1 Algorithme de Peterson

2.2 Algorithme de la boulangerie

2.3 Verrou TAS

2.4 Verrou TATAS

2.5 Verrou TATAS avec attente

2.6 Verrou ticket à attente proportionnelle

2.7 Verrou Anderson

2.8 Verrou Graunke et Thakkar

2.9 Verrou MCS

2.10 Verrou CLH