Parallélisation

1. Applications
2. Historique
3. Parallélisme au niveau du circuit
4. Coprocesseur
5. Multiprocesseur
6. Multi-ordinateur
7. grille

Applications

Simulation numérique : expériences in silico

- Trop grandes : météorologie
- Trop petites : biologie, matériaux – Trop dangereuses : maintenance nucléaire
- Trop couteuses : crash-tests, conception aéronautique
- Impossibles : climat, astrophysique

A réaliser en réalité

- Serveurs traditionnels : Fichiers, Web, Vidéo, Base de données, ...
- Serveurs de calcul : «on demand computing»
- Réalité virtuelle ou augmentée : médias, médical

Peuvent consommer une puissance de calcul et demander une capacité mémoire supérieure à celle d'un processeur

Les débuts du parallélisme

- Multiprogrammation
  - notion de processus
  - système Multics (65)
  - toujours d'actualité, au coeur des UNIX et de NT
- Programmation concurrente
  - les coroutines(Djikstra, 68)
  - multiprogrammation à grain plus fin
  - problèmes de synchronisation
- Parallélisme simulé
  - pas de machine contenant réellement du parallélisme

Parallélisme matériel : première époque

- Processeurs vectoriels (Cray, 1976)
  - circuit spécifique
  - opérations arithmétiques élémentaires (+. ...)
  - données manipulées : vecteurs
  - calcule *n* additions en parallèle au lieu d'une
- contient *n* ALUs au lieu d'une avec un seul microcontroleur pour tous
- Traitement parallèle sur des données
  - même opération sur un ensemble de données
  - le parallélisme se fait sur les données (par opposition aux instructions)
  - "Parallélisme de données"
  - fonctionnement "naturellement" synchrone
- Classification traditionnelle
  - SIMD : Single Instruction (flow) Multiple Data

Parallélisme matériel : seconde époque

- Fin des années 80
  - déclin des architectures tout-propriétaires
  - exemple type : Cray avec le T3D (processeurs ALPHA, réseau haut débit propriétaire)
  - déclin des architectures SIMD : nouveaux types d'applications parallèles envisageables
  - émergence de nouveaux modèles de programmation Orientés instructions
- Tendance actuelle
  - processeurs standards + réseau standard (structure de grappes de machines) : pas trop cher
  - architectures spécifiques (SGI Origin2000) : chers, offrent des fonctionnalités spécifiques tant matérielles que logicielles

Architecture Parallèle

Plusieurs Niveaux de parallèlisme

- Parallélisme au niveau du circuit
- Coprocesseur
- Multiprocesseur
- Multi-ordinateur
- grille

Généralisation de la parallélisation d'exécution d'instructions

- On a vu le système du pipeline : plusieurs instructions simultanées en exécution décalée
- Approches complémentaires :
  - Ajouter des unités (approche superscalaire)
    - Unités de calculs, de commandes ou même pipeline complet
    - Permettre l'exécution complète ou partielle de plusieurs instructions en parallèle
    - Exemple : AMD Athlon 64 : 3 UAL, 3 FPU et 3 unités de décodage d'instructions
  - Mettre plusieurs processeurs sur la même puce : Approche « multi-core »

Parallélisation

- Intérêts de l'ajout d'unités
  - Peut faire plusieurs calculs/traitements en même temps
- Problèmes de l'ajout d'unité
  - Nécessite plus de synchronisation entre unités
  - Prend du temps
  - Pas toujours pleine utilisation de toutes les unités
  - Exemple : on n'a pas forcément des séries de X additions en permanence même après réordonnancement
  - Coût important en terme de transistors et de place
  - Allongement des distances : temps de propagation
  - Coût de fabrication plus important
  - Au final : rarement plus de 2 ou 3 unités d'un type

Calcul vectoriel

- Autre technique : Unités de calcul vectoriel
  - Effectuer un même calcul avec plusieurs valeurs en parallèle (vecteur de 4, 2 ou 1 valeurs)
  - Parallélisme SIMD : Single Instruction, Multiple Data :

	11
+	32

	21	43	72	86
+	34	86	59	12

- Avantages
  - Généralement plus simple et moins lourd de paralléliser
  - En ayant une unité vectorielle avec vecteurs de taille X
  - Que X unités complètes standard non vectorielles

Parallélisation : multi-core

- Mettre plusieurs coeurs de processeurs sur la même puce
- 2 cores dans les approches actuelles : dual-core
- Evolutions futures prévisibles : 4, 8 ... cores par die
- Utilité : faire du multi-processeur mais avec un seul
- Possibilité de multi-processeur sur des cartes mères ayant un seul support physique de processeur
- Au niveau logiciel, permet l'exécution de threads en parallèle
- Plusieurs applications en parallèle
- Plusieurs threads d'une même application

Parallélisation : multi-core

3 approches actuellement chez Intel/AMD:

- Intel Pentium D / XE : approche « basique »
  - Deux processeurs entièrement dupliqués sur la même puce, avec chacun embarquant
  - Unités de commande
  - Unités de calculs
  - Mémoire cache (niveaux L1 et L2)
  - Chaque core communique directement et uniquement avec le chipset via le FSB
  - Pb de performances pour communication entre 2 cores car on passe par le chipset
  - Occupation inutile du FSB avec les communications entre les cores

Parallélisation : dual-core

- Approche AMD Athlon 64 X2
  - 2 cores entièrement dupliqués également mais avec
  - Un bus de communication interne au CPU très rapide entre les 2 cores
  - Améliore les performances de communication entre les 2 cores
  - Approche Intel Core 2 Duo
  - Duplication des 2 cores mais avec un cache L2 commun
  - Meilleure performance pour communication entre 2 cores
  - On passe par le cache directement sans passer par un bus dédié ou le FSB

Parallélisation : dual-core

- Problème dans tous les cas
- Gérer la cohérence des données entre les caches et la mémoire centrale