Ecole d'Automne « Informatique Scientifique pour le Calcul »

–

Architecture des ordinateurs

Françoise Berthoud[1]
Violaine Louvet [2]
Françoise Roch [3]

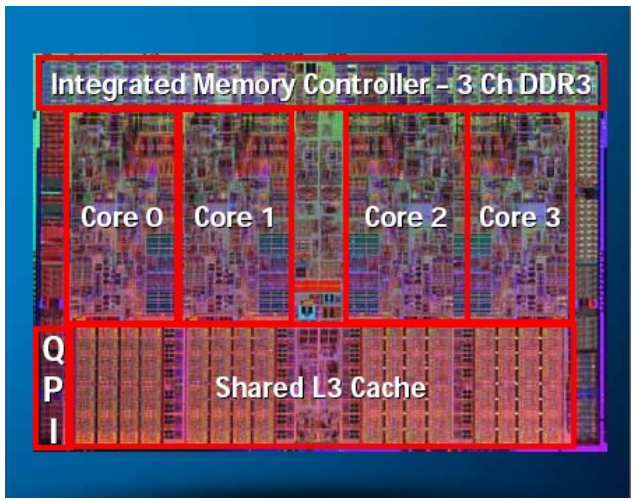
[1]Laboratoire de Physique et de Modélisation des Milieux Condensés - CNRS
[2]Institut Camille Jordan - CNRS
[3]Observatoire des Sciences de l'Univers de Grenoble

29/09/08 - 03/10/08

Introduction

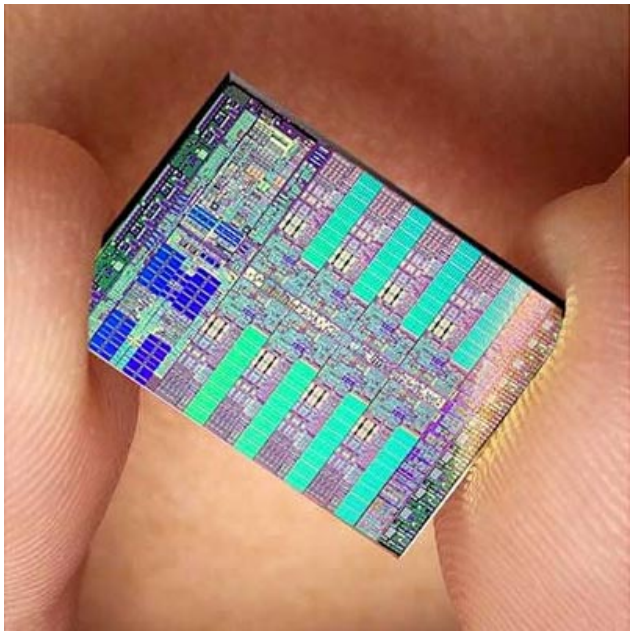
Décoder la relation entre l'architecture et les applications :

- Adapter les algorithmes, la programmation,
- Comprendre le comportement d'un programme,
- Choisir son infrastructure de calcul en fonction de ses besoins.



Nehalem

GeForce 8800



Cell

Sommaire

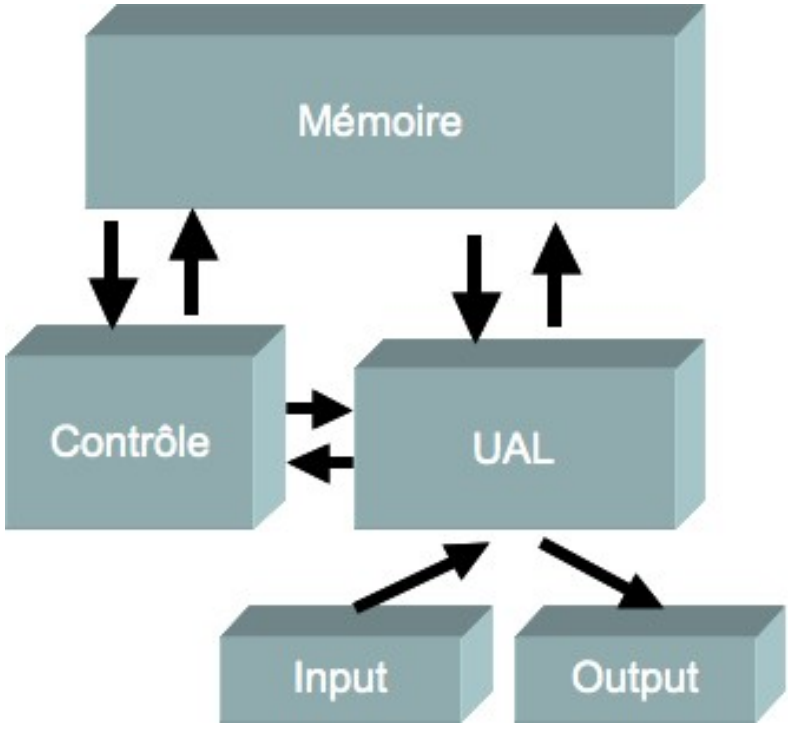
Outline

Evolution des architectures

- Loi de Moore : nb de transistors par circuit intégré $\times$ 2 tous les 2 ans.
- En 10 ans, la finesse de gravure est passée de $0,25\ \mu m$ (Pentium III) à $0,045\ \mu m$ (Core 2 Penryn).
- Evolution de l'architecture des processeurs (pipeline, duplication des unités fonctionnelles, exécution spéculative, exécution désordonnée ...).
- Evolution des interactions architecture/applications : dans le cas de l'Itanium, le compilateur fait partie intégrante de l'architecture.
- Evolution des infrastructures vers un haut degré de parallélisme en intégrant un grand nombre de processeurs.

Top 500 Bench Linpack	**Puissance soutenue en Gflops**	**Puissance crête en Gflops**	**Nombre de processeurs ou de cœurs**
Juin 1993	59.7	131	1024
Juin 2008	1 026 000	1 375 780	122 400

Modèle de Von Neumann (1945)



- La mémoire : contient le programme (instructions) et les données.
- Une Unité Arithmétique et Logique : UAL qui effectue les opérations.
- Une unité de contrôle chargée du séquençage des opérations.
- Une unité d'Entrée/Sortie.

Classification de Flynn

Classification des architectures selon 2 dimensions :

- instruction,
- data.

Deux états possible pour chacune des dimensions :

- single,
- multiple.

SISD Single Instruction, Single Data	SIMD Single Instruction, Multiple Data
MISD Multiple Instruction, Single Data	MIMD Multiple Instruction, Multiple Data

Outline

Comment faire des processeurs plus rapides ?

1. Augmenter la fréquence d'horloge (limites techniques, solution coûteuse).
2. Permettre l'exécution simultanée de plusieurs instructions :
   - Instruction Level Parallelism : pipelining, superscalabilité, architecture VLIW et EPIC.
   - Thread Level Parallelism : multithreading et SMT.
3. Améliorer les accès mémoire.

Outline

Augmenter la fréquence d'horloge

- La fréquence d'horloge détermine la durée d'un cycle.
- Chaque opération utilise un certain nombre de cycles.
- La fréquence d'horloge est fonction de :
  - la technologie des semi-conducteurs,
  - le packaging,
  - les circuits.

Les limites :

- La chaleur dégagée par le processeur est fonction de sa fréquence d'horloge.
- Le nb de cycles d'horloge pour interruptions, « cache miss »ou mauvaise prédiction de branchement augmentent.

Outline

Exécution en parallèle de plusieurs instructions

1 CPU = plusieurs unités fonctionnelles qui peuvent travailler en parallèle :

- une unité de gestion des bus (unité d'entrées-sorties) en interface avec la mémoire vive du système,
- une unité d'instruction (control unit) qui lit les données arrivant, les décode et les envoie à l'unité d'exécution,
- une unité d'exécution qui accomplit les tâches que lui a données l'unité d'instruction, composée notamment de :

  une ou plusieurs unités arithmétiques et logiques (UAL) qui assurent les fonctions basiques de calcul arithmétique et les opérations logiques.

  une ou plusieurs unités de virgule flottante (FPU) qui accomplissent les calculs complexes. L'instruction de calcul de base est la multiplication / addition (FMA pour Floating-point Multiply and Add) en double précision.

La performance crête dépend de la taille des registres et est fonction de la précision (SP,DP). Cette mesure de performance (GFlops) n'évalue que les opérations sur les flottants, on peut utiliser les MIPS pour mesurer une performance sur le nombre d'instructions par seconde.

Performance crête du processeur Xeon (quadcore)

Il comprend 2 FPU et 1 FMA : 4 opérations flottantes/cycle $\times$ 2.5 GHz $\times$ 4 = 40 GFlops / proc (en DP)

Exécution simultanée de plusieurs instructions : ILP (pipelining)

Principe

Une opération s'exécute en plusieurs étapes indépendantes par des éléments différents du processeur. Ces étapes s'exécutent simultanément sur des données distinctes (parallélisme d'instructions).

Phases d'exécution d'une instruction :

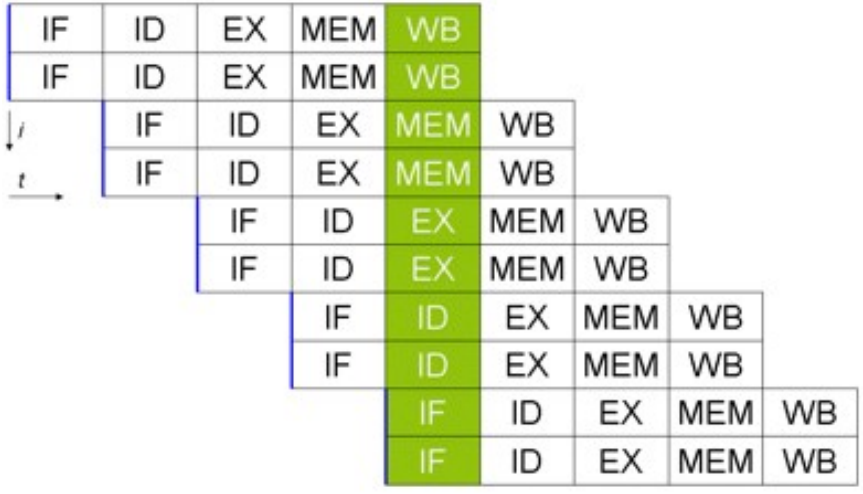
- IF : Instruction Fecht
- ID : Instruction Decode/Register Fecht
- EX : Execution/Effective Address
- MA : Memory Access/ Cache Access
- WB : Write-Back

5 instructions en parallèle en 9 cycles (25 cycles en séquentiel) $\longrightarrow$ permet de multiplier le débit avec lequel les instructions sont exécutées par le processeur.

Instruction Level Parallelism : architectures superscalaires

Principe

Duplication de composants (FMA, FPU) et exécution simultanée de plusieurs instructions.



10 instructions en 9 cycles

- Gestion des instructions :

  Statique (in-order) : exécutées dans l'ordre du code machine.
  Dynamique (out-of-order) : le hardware modifie l'ordre des instructions pour favoriser le parallélisme.

- Exécution spéculative : faire une hypothèse sur la suite d'un traitement après un branchement conditionnel (lorsque la valeur de la condition n'est pas encore calculée).

Les problèmes de dépendances entre instructions :

- Partage des ressources : par exemple la mémoire.
- Dépendances des données entre instruction : une instruction produit un opérande utilisé immédiatement par l'instruction suivante.
- Dépendances des contrôles : une instruction est une branche.

Appliquer la même instruction à plusieurs données : SIMD

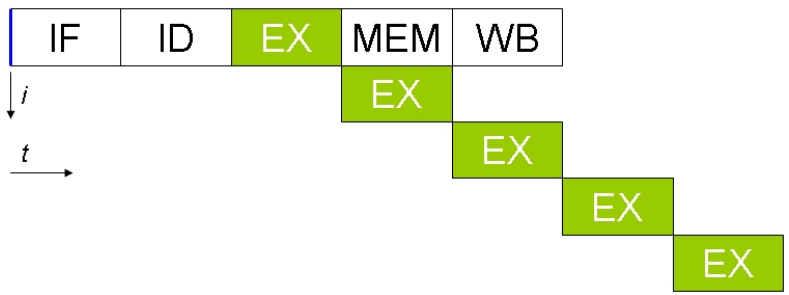
Le mode SIMD (Single Instruction on Multiple Data) permet d'appliquer la même instruction simultanément à plusieurs données (stockées dans un registre) pour produire plusieurs résultats.

Exemples de jeux d'instructions SIMD

- MMX, MultiMedia eXtensions : permettent d'accélérer certaines opérations répétitives dans des domaines tels que le traitement de l'image 2D, du son et des communications.
- SSE, Streaming SIMD Extensions (SSE2, SSE3, SSE4 en 2008) : par exemple instructions pour additionner et multiplier plusieurs valeurs stockées dans un seul registre (registre SSE).
- 3DNow : jeu d'instructions multimédia développé par AMD.

Les architectures vectorielles

- De nombreux problèmes sont intrinsèquement vectoriels : ils opèrent sur des vecteurs de données unidimensionnels.
- Certaines architectures disposent d'instructions vectorielles :
  - l'instruction vectorielle est décodée une seule fois, les éléments du vecteur sont ensuite soumis un à un à l'unité de traitement.
  - Les pipelines des unités de traitement sont pleinement alimentés.
- Exemples : Cray, NEC SX, Fujitsu VPP, Altivec (PowerPC G4 et G5)



Les limites du parallélisme d'instructions/données

Obstacles à la performance sur les systèmes superscalaires standard :

- les branchements (IF, CASE, ...) :
  - les mauvaises prédictions : en cas d'erreur de prédiction, il faut revenir en arrière,
  - les petites branches ont peu de code à exécuter ce qui limite le parallélisme.
- La latence de la mémoire :
  - utilise plus d'un cycle d'horloge.
- L'extraction du parallélisme des instructions :
  - le compilateur sérialise le code, dont le parallélisme intrinsèque doit être redécouvert dynamiquement par le processeur.

Dépasser les limites en « aidant »le processeur

Les architectures VLIW (Very Long Instruction Word) et EPIC (Explicitly Parallel Instruction Set Computing) permettent de traiter des instructions longues, agrégations d'instructions courtes indépendantes de façon à les exécuter explicitement en parallèle.

- VLIW : Le compilateur a la charge d'organiser correctement les instructions parmi les bundles, tout en respectant les types de dépendances habituelles qui sont normalement gérées au niveau matériel par les architectures classiques.

  $\longrightarrow$ On gagne en performance (pas de contrôle), mais la compatibilité binaire entre générations successives est quasi-impossible. Le code généré est spécifique à une implémentation de l'architecture.

Dépasser les limites en « aidant »le processeur

- EPIC : type d'architecture de microprocesseurs utilisée notamment dans les Itaniums. Le parallélisme est exprimé de manière indépendante de la mise en oeuvre du processeur. Disparition du réordonnancement à l'exécution : les instructions sont exécutées dans l'ordre exact dans lequel le compilateur les a mis.

  $\longrightarrow$ l'effort d'optimisation repose sur le compilateur qui a la charge d'organiser statiquement les dépendances inter-instructions.

Exemple

Sur un Itanium, le flot d'instruction est décomposé en blocs (ou bundles, de taille 128 bits) de 3 instructions successives. Chaque bundle contient également un template spécifiant les dépendances au sein du bloc ou entre blocs successifs. Les blocs dépendants les uns des autres forment des groupes. Les instructions appartenant au même groupe peuvent être exécutées en parallèle en fonction du nombre d'unités fonctionnelles disponibles.

Outline

Améliorer le remplissage du pipeline du processeur : TLP

Idée : mixer deux flux d'instructions arrivant au processeur pour optimiser l'utilisation simultanée de toutes les ressources (remplir les cycles perdus - cache miss, load ...).

Le SMT (Simultaneous Multithreading) partage du pipeline du processeur entre plusieurs threads(d'un même programme ou de deux programmes différents). Les registres et les caches sont aussi partagés.

L'hyperthreading = SMT d'Intel.

Le multithreading dégrade les performances individuelles des threads mais améliore les performances de l'ensemble.

Type d'approche Multithreads	Ressources partagées entre threads	Mécanisme d'ordonnancement
Aucun	Tout	Géré explicitement por l'OS
A grain fin	Tout sauf les fichiers de registres et d'état/contrôle	switch à chaque cycle
A grain grossier	Tout sauf I-fetch buffers, register file et control logic/state	switch à chaque état d'attente du pipeline
SMT	Tout sauf I-fetch buffers, return address stack, architected register file, control logic/state, reorder buffer, store queue, etc.	Tous les contextes sont actifs de façon concurrente
CMP	Cache L2, interconnexion système	Tous les contextes sont actifs de façon concurrente

Outline

Outline

Les différentes technologie de mémoire

- Principalement deux types à base de semi-conducteur :

  DRAM (Dynamic Random Access Memory) chaque bit est représenté par une charge électrique qui doit être rafraîchie à chaque lecture/écriture.

  SRAM (Static Random Access Memory) retient ses données aussi longtemps qu'il y a du courant.
- Temps d'un cycle SRAM de 8 à 16 fois plus rapide qu'un cycle DRAM.
- Coût de la mémoire SRAM de 8 à 16 fois plus élevé que la mémoire DRAM.

Solution la plus couramment choisie :

1. De 1 à 3 niveaux de SRAM en mémoire cache.
2. La mémoire principale en DRAM.
3. La mémoire virtuelle sur des disques

Type	Taille	Vitesse	Coût/bit
Registre	< 1 KB	< 1 ns	$$$$
SRAM On-chip	8 KB - 6 MB	< 10 ns	$$$
SRAM Off-chip	1 MB - 16 MB	< 20 ns	$$
DRAM	64 MB - 1 TB	< 100 ns	$
Flash	64 MB - 32 GB	< 100 μs	c
Disk	40 GB - 1 PB	< 20 ms	$\sim$0

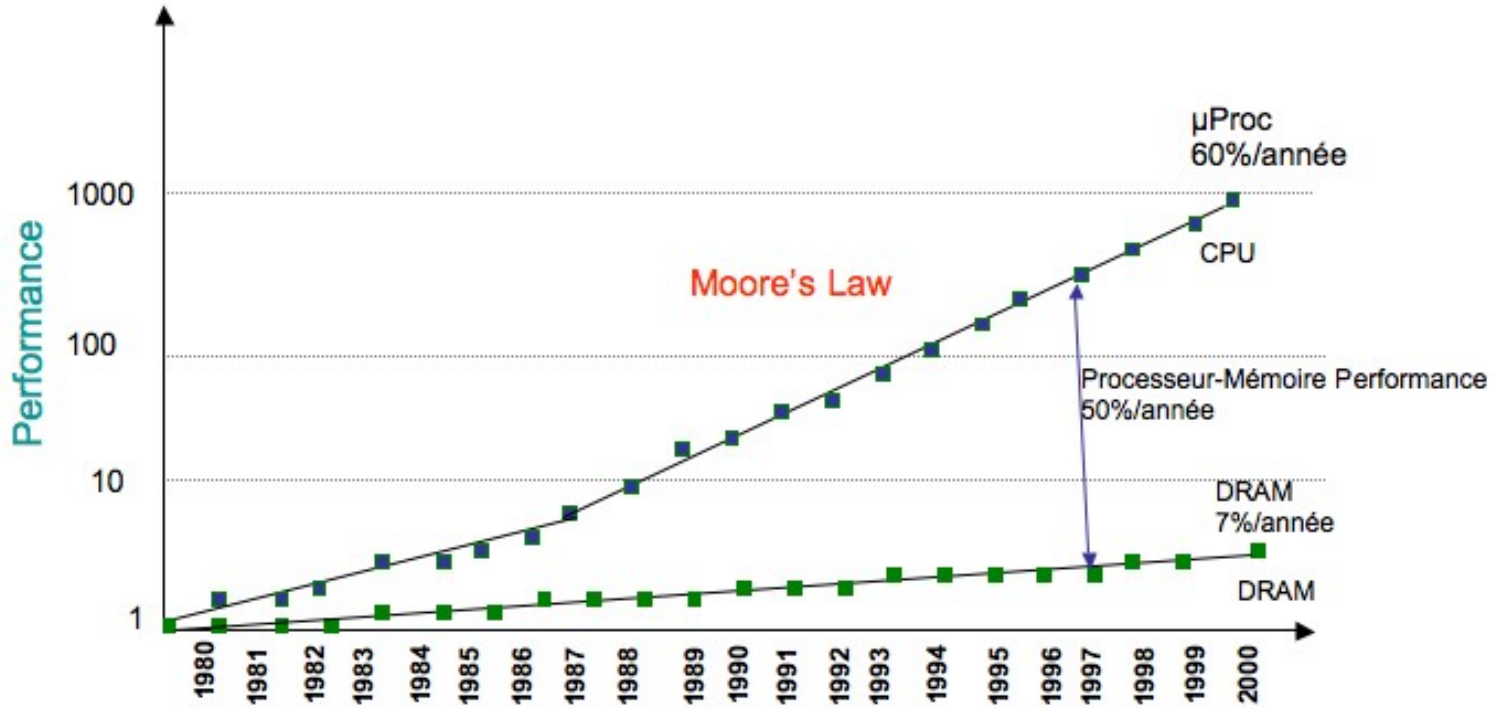
Outline

Hiérarchisation de la mémoire

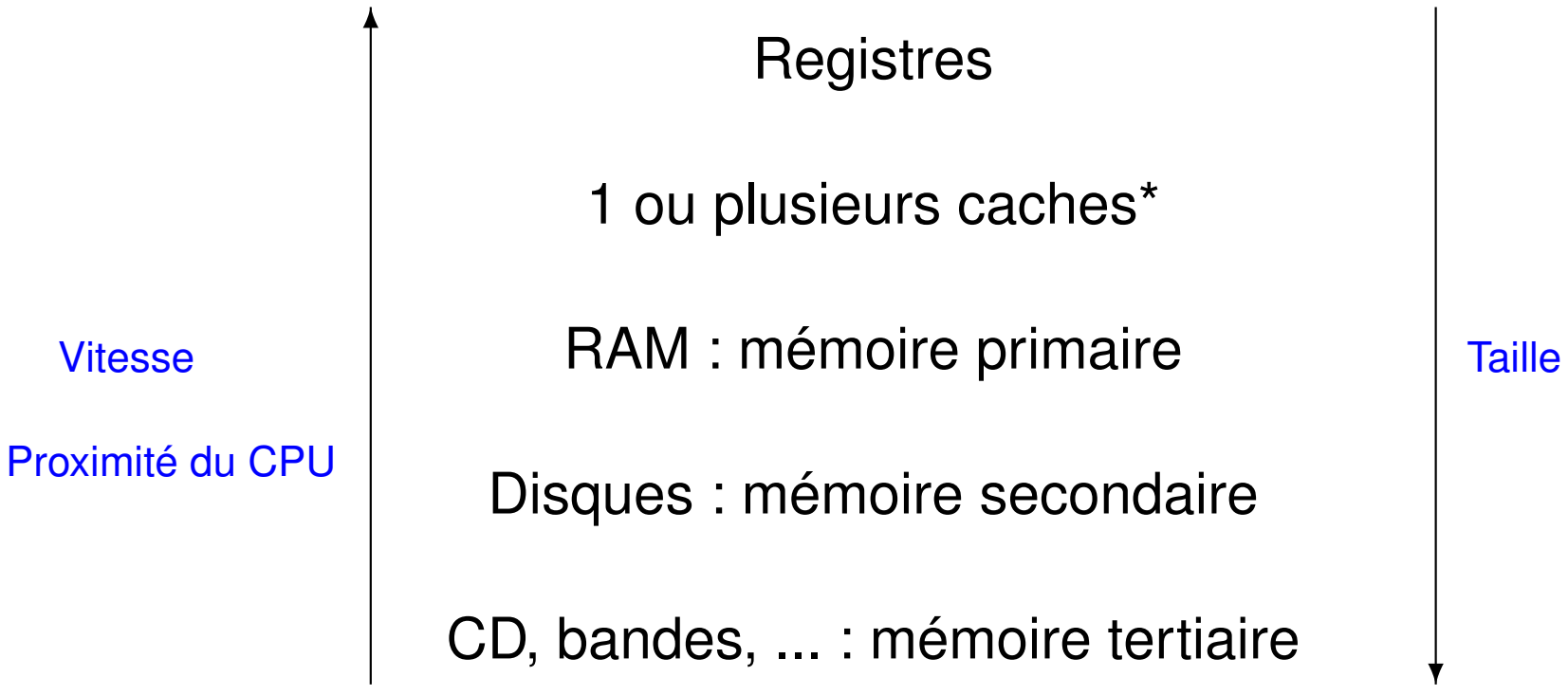
Cas idéal : Mémoire la plus grande et la plus rapide possible

La performance des ordinateurs est limitée par la latence et la bande passante de la mémoire :

- Latence = temps pour un seul accès.
  Temps d'accès mémoire $\gg$ temps cycle processeur.
- Bande passante = nombre d'accès par unité de temps.



Hiérarchisation de la mémoire



* les caches peuvent être organisés de façon hiérarchiques

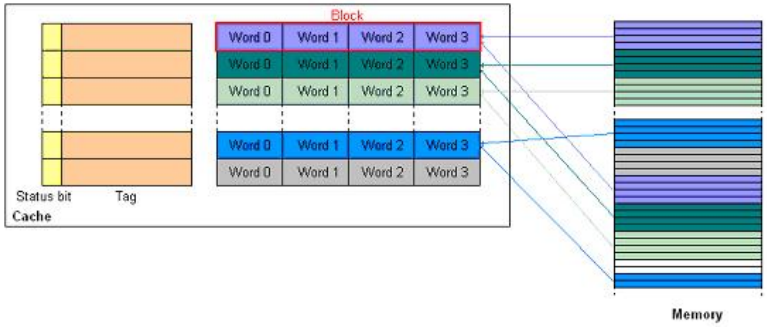
Outline

Organisation des caches

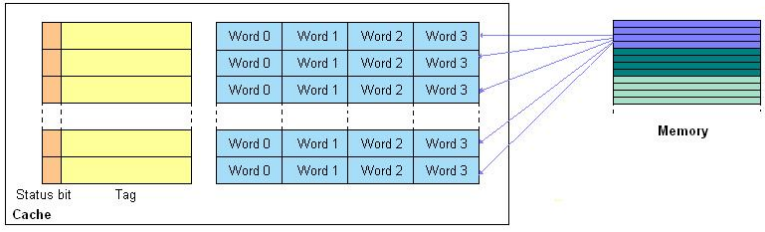
- Le cache est divisé en lignes de n mots.
- 2 niveaux de granularités :
  - le CPU travaille sur des « mots » (par exemple, 32 ou 64 bits).
  - Les transferts mémoire se font par blocs (par exemple, lignes de cache de 256 octets).
- Une même donnée peut être présente à différents niveaux de la mémoire : problème de cohérence et de propagation des modifications.
- Les lignes de caches sont organisées en ensembles à l'intérieur du cache, la taille de ces ensembles est constante et est appelée le degré d'associativité.

Organisation des caches

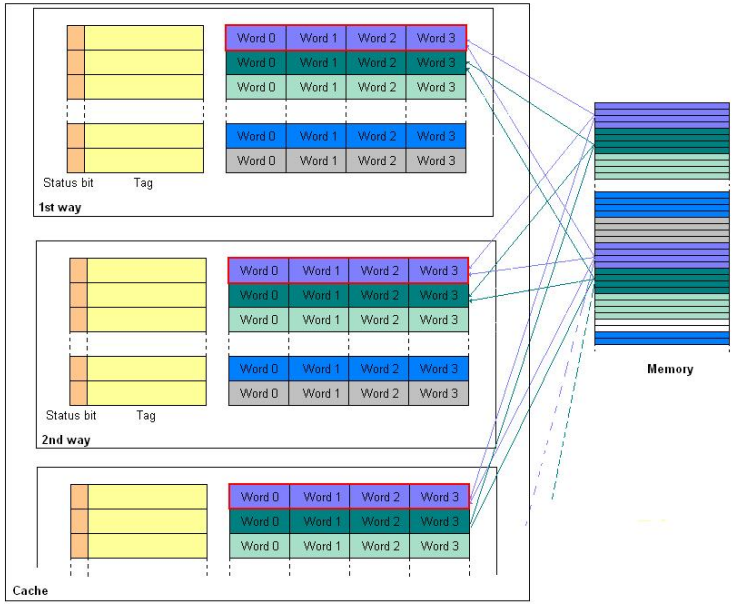
Direct Map : chaque adresse mémoire est associée à une ligne de cache déterminée (degré d'associativité = 1).



Fully associative : chaque adresse mémoire correspond à n'importe quelle ligne de cache.



k way associative : chaque adresse a une alternative de *k* lignes.



Outline

Notion de localité mémoire

- Localité temporelle : si une zone est référencée, elle a des chances d'être référencée à nouveau dans un futur proche.
  - Exemple de proximité temporelle : dans une boucle simple, chaque itération accède aux mêmes instructions.
- Localité spatiale : si une zone est référencée, les zones voisines ont des chances d'être référencées dans un futur proche.
  - Exemple de proximité spatiale : dans un bloc simple, chaque instruction sera accédée l'une après l'autre.

Proximité temporelle

```
DO I = 1,10 0000
        S1 = A(I)
        S2 = A(I+K) # S2 sera réutilisé à l'itération I+K
END DO
```

- L'idée de base est de conserver A(I+K) (lecture de la référence à l'itération I, déclaration S2) en mémoire rapide jusqu'à sa prochaine utilisation à l'itération I+K, déclaration S1.
- Si on veut exploiter la localité temporelle via les registres, cela suppose qu'il faille au moins K registres.
- Cependant, dans le cas général, on aura certainement besoin de stocker d'autres données.

L'évaluation précise de la proximité temporelle est très complexe

Proximité spatiale

- Le voisinage d'une zone localisée par une adresse doit être évaluée dans l'espace d'adressage virtuel.
- Les structures de données sont linéarisées et réduites à une dimension pour être placées dans l'espace d'adressage linéaire.

Exemple en Fortran : rangement des tableaux 2D par colonne

```
DO I = 1,10000
      DO J = 1,1000
            X1 = A(I,J)
            X2 = B(J,I)
      END DO
END DO
```

- Pour A : localité spatiale parfaite, accès à des zones de mémoire consécutives.
- Pour B : 2 références consécutives sont distantes de 1000 emplacements mémoires.

Pour exploiter la localité spatiale, il faut que la distance entre 2 références soit inférieure à la taille de la ligne du cache.

Outline

Lecture/Ecriture des données

Cache hit

Lecture

- fournit la donnée à partir du cache

Politiques d'écriture :

- **write through** : écriture à la fois dans le cache et la mémoire (simplifie la cohérence) ;
- **write back** : écriture seulement dans le cache, la mémoire est mise à jour une fois la ligne de cache vidée.

Cache miss

Lecture

- récupération de la donnée en mémoire principale
- stockage de cette donnée dans le cache
- fourniture de la donnée à partir du cache

Politique d'écriture :

- **no write allocate** : écriture seulement en mémoire principale ;
- **write allocate** : écriture dans le cache.

Les combinaisons usuelles : « write through »et « no write allocate », « write back »et « write allocate ».

Effets des paramètres des caches sur les performances

- Cache plus grand
  - ☺ réduit les « cache miss »,
  - ☹ augmente le temps d'accès.
- Plus grand degré d'associativité
  - ☺ réduit les conflits de cache,
  - ☹ peut augmenter le temps d'accès.

Prefetching

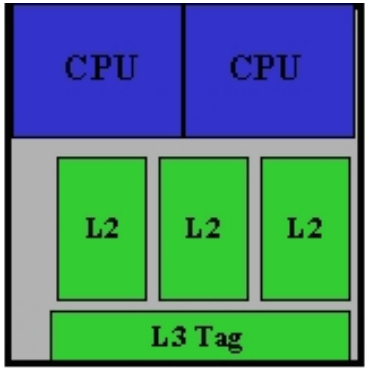
Le prefetching consiste à anticiper les données et les instructions dont le processeur aura besoin, et ainsi les précharger depuis la hiérarchie mémoire (le L2 ou la mémoire centrale).
Différents type de prefetching :

- mécanisme hardware de préchargement,
- mécanisme software de préchargement,
- mécanisme mixte.

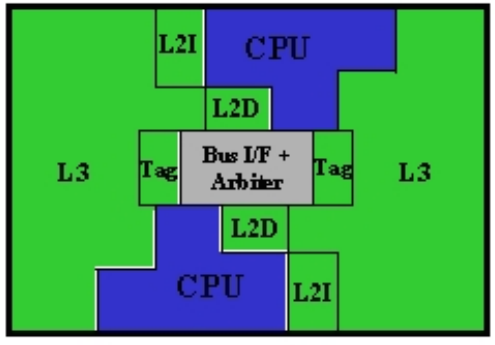
Le prefetching n'est pas gratuit, il consomme de la bande passante qui est une ressource critique.

Quelques exemples

	POWER 5	Montecito
L1 I cache	64 kb, 2 way associative, 4 cycles latency	16 kb, 4 way associative, 1 cycle latency
L1 D cache	32 kb, 4 way associative, 4 cycles latency	16 kb, 4 way associative, 1 cycle latency
L2 cache	1.92 MB, 10 way associative, 13 cycles latency	2×1 MB (I), 2×256 kb (D), 8 way associative, 5 cycles latency
L3 cache	36 MB, 12 way associative, 87 cycles latency	24 MB, 2 way associative, 14 cycles latency
Memory	16 GB/s per device, 220 cycles latency	21.3 GB/s per device



POWER5
389 mm²



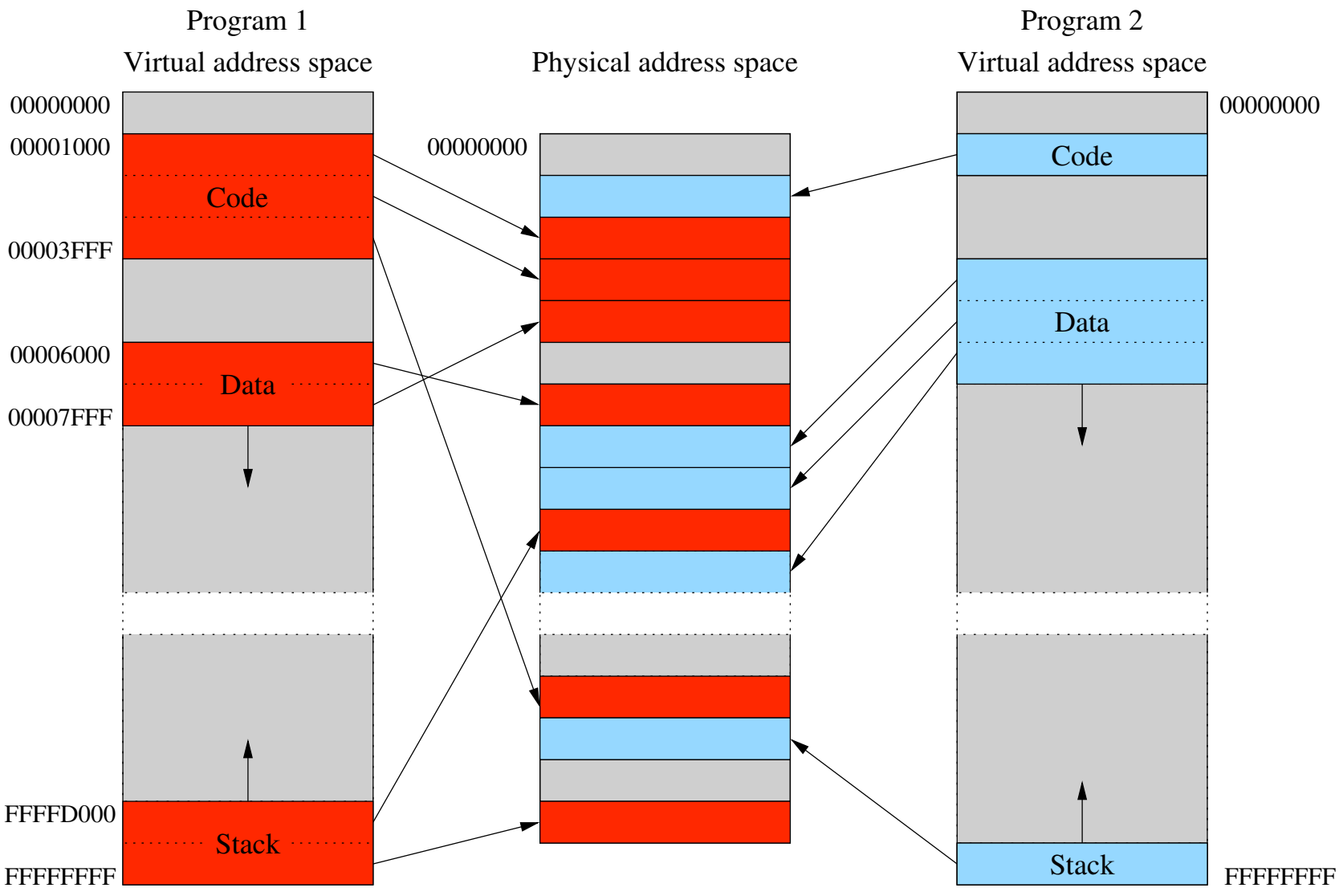
Montecito
~580 mm² (est)

Outline

Mémoire virtuelle

- La mémoire virtuelle est le dernier niveau de la hiérarchie mémoire.
- Espace d'adressage virtuel (ou logique) :
  - Quantité maximale d'espace d'adressage disponible pour une application.
  - Cet espace d'adressage virtuel est en correspondance avec l'espace d'adressage physique pour l'accès aux données via le MMU (Memory Management Unit) au niveau hard, et via le système d'exploitation.
- L'espace d'adressage virtuel de chaque programme et l'espace mémoire sont divisés en pages (physiques ou virtuelles) de même taille.

Correspondance espace d'adressage physique - espace d'adressage virtuel



Accès aux pages de l'espace physique

- Taille des pages :

  X86 pages de 4KB (on peut aussi trouver 2 MB, 4 MB).

  NEC SX supporte à la fois des petites et des grandes pages.

  Itanium plusieurs tailles de page possibles : 8 KB, 16KB, 64 KB, 1 MB, ...

- Le placement est « fully associative ». Une table des pages contient l'ensemble des informations nécessaires à la correspondances adresses virtuelles <-> adresses physiques.
- Problème : 4 GB de mémoire et des page de 4 KB impliquent une table de 1 millions d'entrées !

  $\longrightarrow$ Table de pages multiniveaux,

  $\longrightarrow$ la table peut être mise en cache.

Outline

Page faults

L'adresse virtuelle référence une page qui n'est pas trouvée en mémoire physique :

- Si la mémoire physique est pleine, virer de la mémoire physique une page (remplacement) :
  - choisir une page « victime »,
  - si elle a été modifiée, la réécrire sur disque,
  - modifier les indicateurs de présence dans la table.
- Dans tous les cas :
  - charger la page référencée en mémoire physique (placement),
  - modifier les indicateurs de présence dans la table.

Les défauts de page sont extrêmement coûteux !

Outline

TLB, Translation Lookaside Buffer

Objectif : Eviter l'accès multiple au cache pour récupérer les adresses physiques.
Idée : utiliser un cache spécifique, le TLB, qui ne référence que les pages virtuelles les plus récentes.

- La taille du TLB est limitée.
- TLB misses : l'adresse n'est pas dans le TLB :
  - le processeur parcourt la table des pages du processus, si la page est dans la table, chargement des informations relatives à cette page dans le TLB. Il faut compter quelques dizaine de cycle pour trouver et charger les infos dans le TLB.
  - Si la page n'est pas en mémoire principale, c'est un défaut de page. Il faut compter plusieurs centaines de milliers de cycles pour récupérer un défaut de page.
- Eviter les défauts de TLB :
  - Eviter des accès aléatoires à des ensembles de données de plus de 1 Mo de façon à eviter les "TLB miss".
  - Le TLB est « 2 way-associative »(2 voies de 128), il faut donc éviter des sauts (strides) de taille une grande puissance de 2 pour éviter les TLB miss.

Outline

Outline

Communications CPU - Mémoire - IO

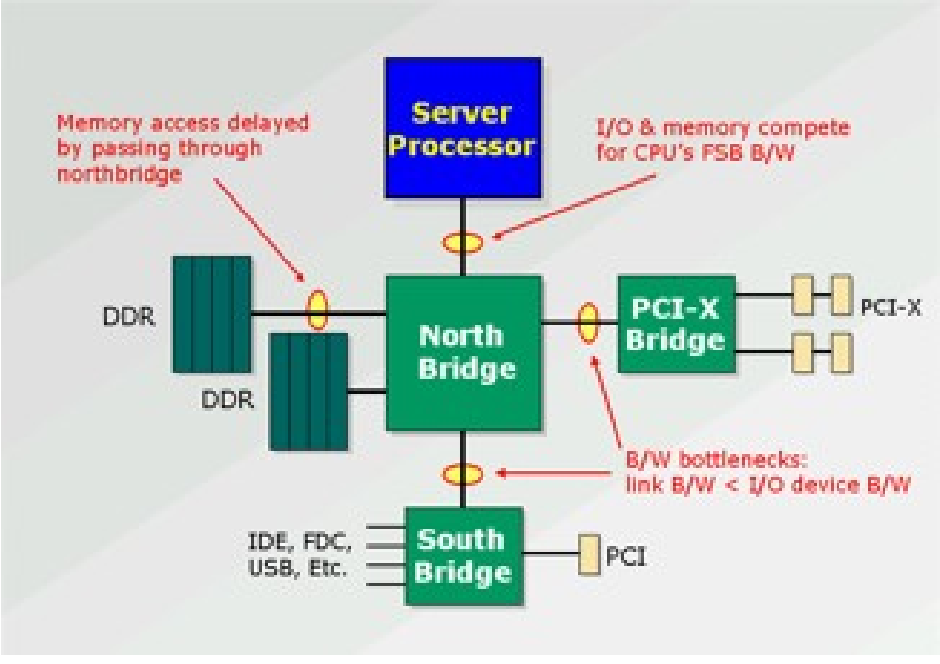
Front Side Bus

- Hypertransport, solution AMD
  - Chaque processeur dispose de 2 canaux de mémoire (contrôleur intégré au processeur).
  - Les processeurs sont reliés entre eux via un lien hypertransport à 2 GHz en HT 2.0 (passage à 3.2 GHz en HT 3.1).
- QuickPath Interconnect (QPI), solution Intel
  - Remplace le FSB sur l'architecture Nehalem, similaire à l'HT.
  - Le principal intérêt du bus QPI provient de sa topologie point à point : le bus connectant les processeurs au chipset n'est plus partagé.

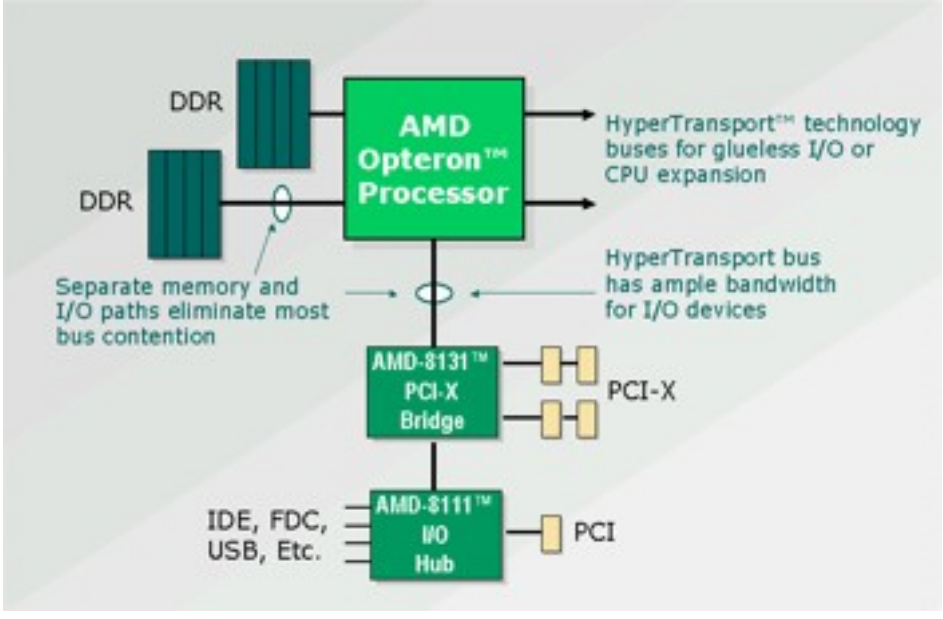
Limite de la bande passante sur hypertransport (AMD, V3 en 2.6 GHz) : 20.8 GB/s

(cf `http://en.wikipedia.org/wiki/List_of_device_bandwiths`).

Exemple : xeon versus opteron



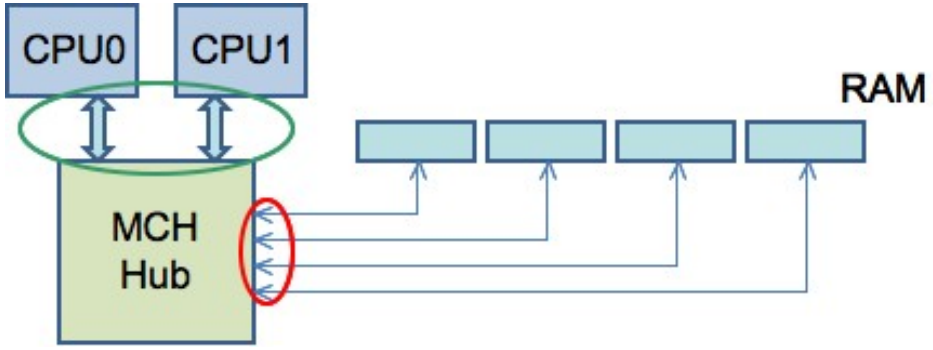
Pentium 4



Opteron

Communications CPU - Mémoire

Pour évaluer la bande passante entre CPU et mémoire, il faut connaître la fréquence du FSB.
Sur la gamme intel : augmentation régulière de cette fréquence, actuellement 1 600 MHz.



Exemple : calcul de la bande passante mémoire locale sur un noeud bi-sockets Intel Hapertown :

- Bande Passante au niveau de l'interface processeurs : sur une carte bi-sockets 2 FSB 64 bits à 1 600 MHz, soit $2 \times 1600 \times 8 = 25.8$ GB/s.
- Bande Passante au niveau de l'interface mémoire : la mémoire est adressée via 4 canaux à 800 MHz (si FBDIMM 800 MHz). La largeur du bus est de 72 bits, dont 8 bits de contrôle, soit 64 bits de données. Ce qui donne : $4 \times 800 \times 8 = 25.6$ GB/s.

Outline

Bus interne d'extension

- Bus série full duplex PCI-Express
  - Remplace le PCI-X qui utilise un unique bus 32-bit bidirectionnel alterné (half duplex).
  - Utilise une interface série (1 bit donc) à base de lignes bidirectionnelles.

Limite de la bande passante sur PCI-X : 64 bits $\times$ 133 MHz = 1056 MB/s
Limite de la bande passante sur PCI-Express (en V2 : 500 MB/s par canal) :
2X : 1 GB/s 4X : 2 GB/s

Outline

Outline

Les architectures multicœurs

- Un processeur composé d'au moins 2 unités centrales de calcul sur une même puce.
- Permet d'augmenter la puissance de calcul sans augmenter la fréquence d'horloge.
- Et donc réduire la dissipation thermique.
- Et augmenter la densité : les coeurs sont sur le même support, la connectique qui relie le processeur à la carte mère ne change pas par rapport à un mono-cœur.

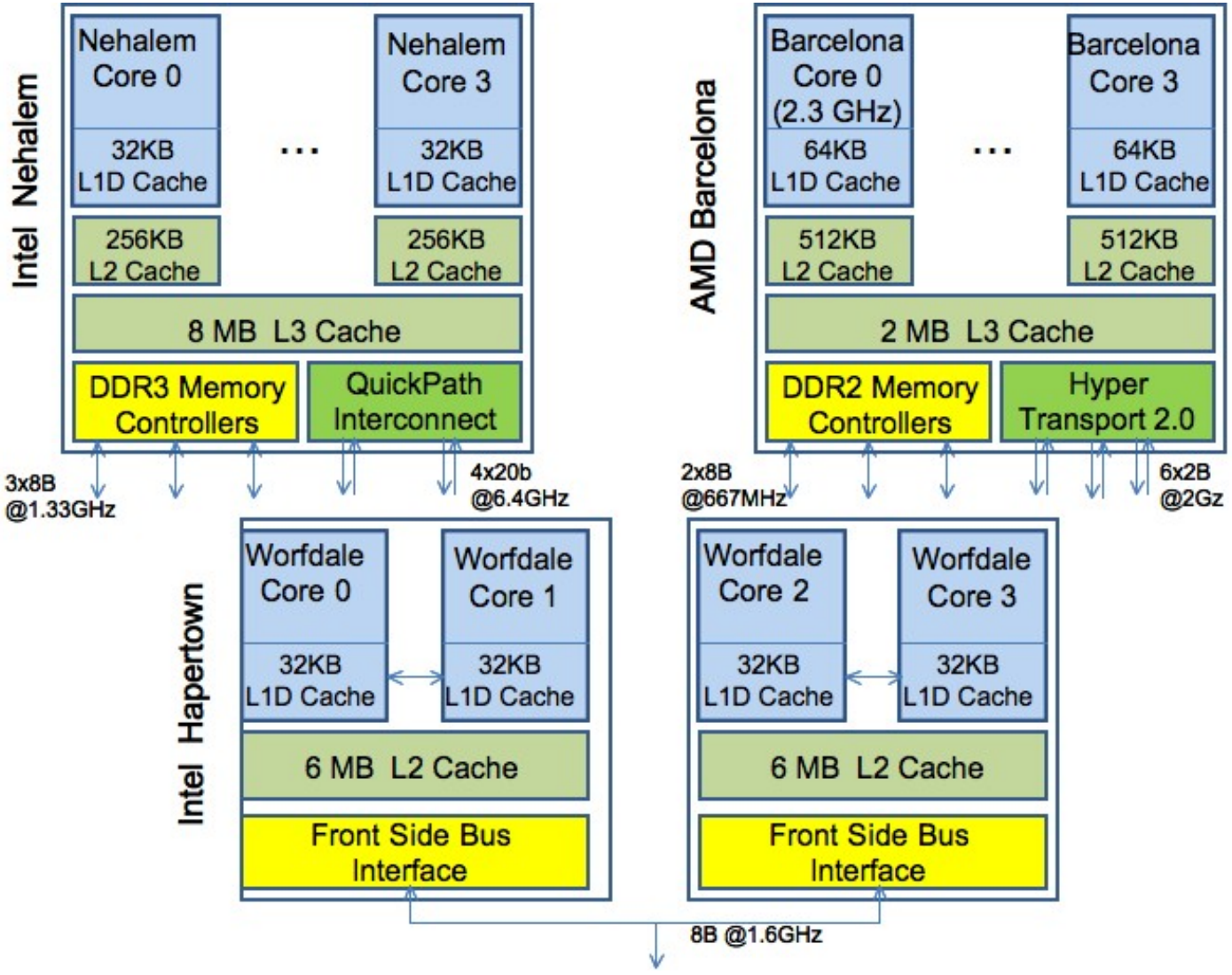
La version originale de la loi de Moore dit que le nombre de transistors le nombre de transistors des circuits intégrés double tous les deux ans. Que faire de tous ces transistors ?

- une complexité sur l'« out of order »accrue,
- des caches de plus en plus grands (mais qui limitent les progrès au niveau de la vitesse),
- plus de CPUs.

Exemples

POWER5, POWER6, Intel Core 2 duo, xeon 53XX (quad-cœurs Clovertown, Hapertown), AMD opetron double cœurs denmark, quad -cœurs Barcelona, Montecito...

Exemples de processeurs multicœurs



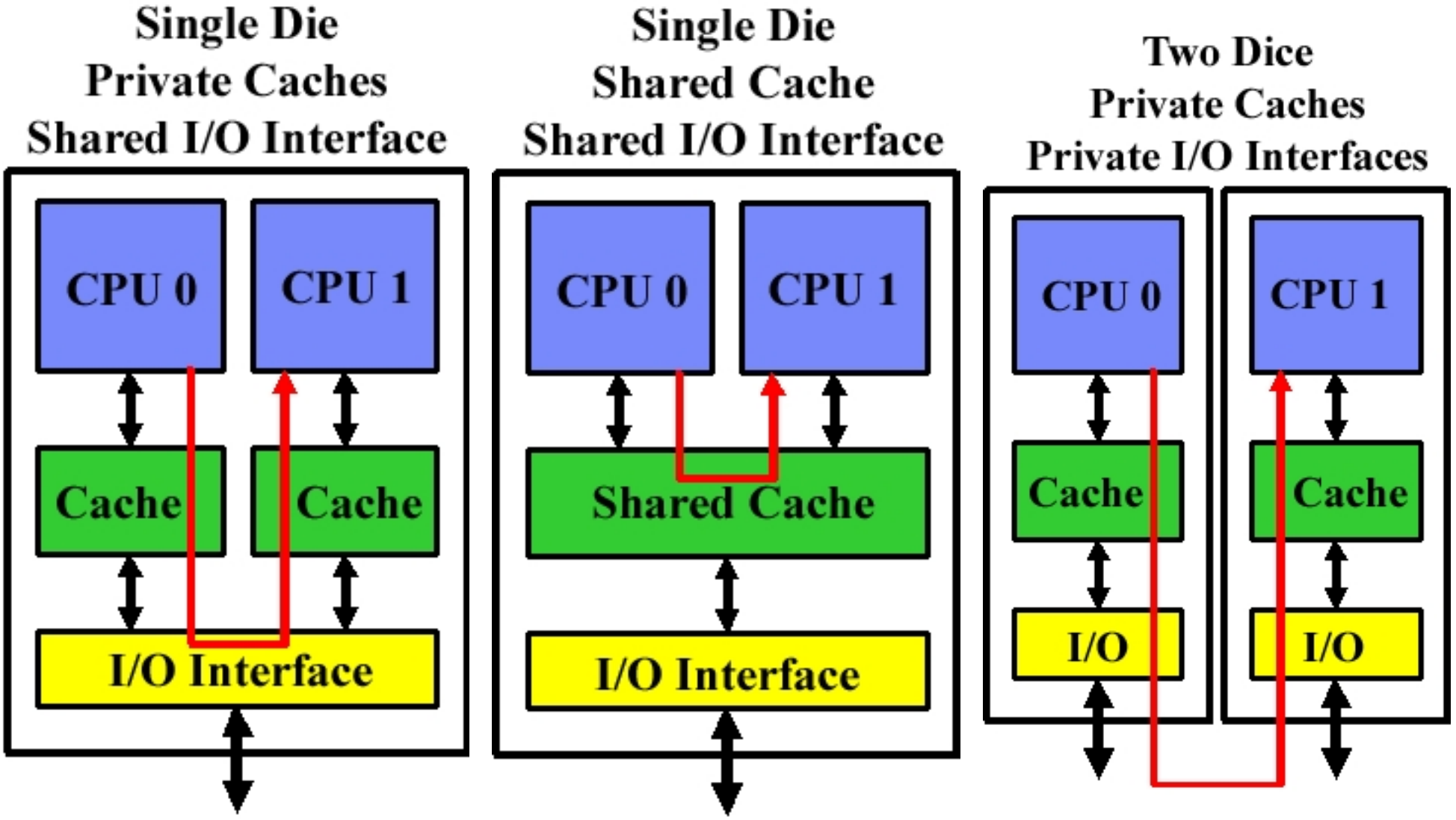
Pourquoi les multicœurs ?

Quelques ordres de grandeur

	Single Core	Dual Core	Quad Core
Core area	A	$\sim$ A/2	$\sim$ A/4
Core power	W	$\sim$ W/2	$\sim$ W/4
Chip power	W + O	W + O’	W + O”
Core performance	P	0.9P	0.8P
Chip performance	P	1.8P	3.2P

Outline

Organisation des caches



Partage des caches L2 et L3

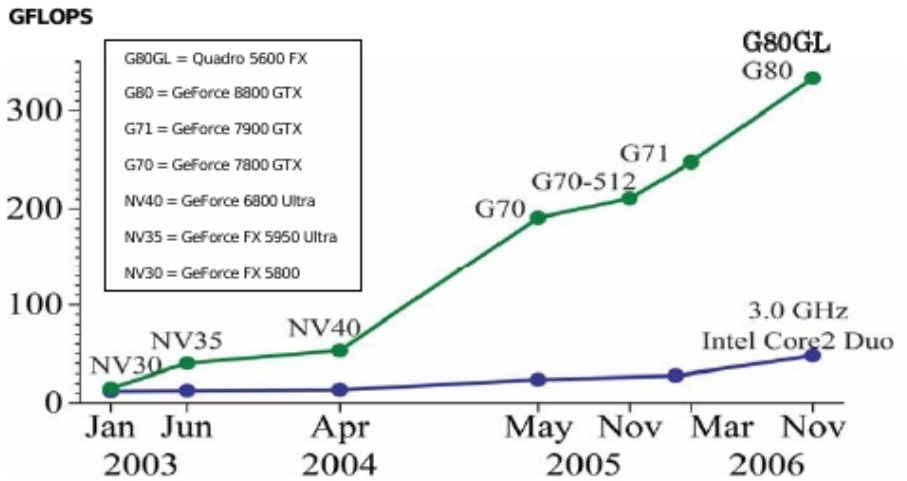
- Partage du cache L2 (ou L3) :
  - ☺ communications plus rapides entre cœurs,
  - ☺ meilleure utilisation de l'espace,
  - ☺ migration des threads plus facile entre les cœurs,
  - ☹ contention au niveau de la bande passante et de la mémoire,
  - ☹ problème de cohérence.
- Pas de partage entre les caches :
  - ☺ pas de contention,
  - ☹ communication/migration plus coûteuse, passage systématique par le cache local.
- La tendance :
  - cache L2 non partagé, cache L3 partagé (IBM Power5+ / Power6, Interl Hapertown / Nehalem).

Outline

Outline

Processeurs spécialisés : les GPU

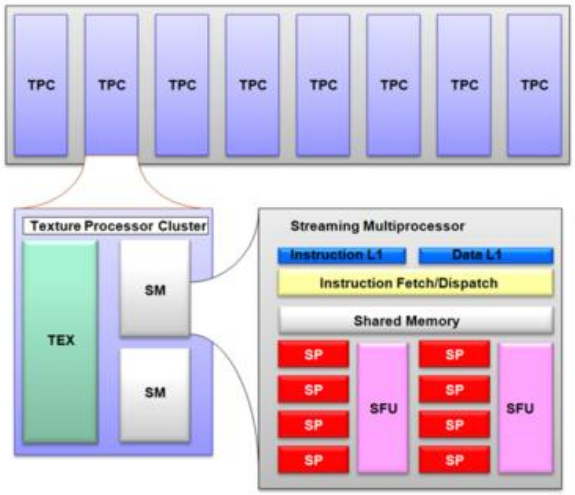
GPU = Graphic Processor Unit



- Performance théorique GeForce 8800 vs Intel Core 2 Duo 3.0 GHz : 367 GFlops / 32 GFlops.
- Bande passante mémoire : 86.4 GB/s / 8.4 GB/s.
- Présents dans tous les PC : un marché de masse.
- Adapté au massivement parallèle (milliers de threads par application).
- Jusqu'à récemment, uniquement programmable via des APIs graphiques. Aujourd'hui, des modèles de programmation disponibles : CUDA (Compute Unified Device Architecture).

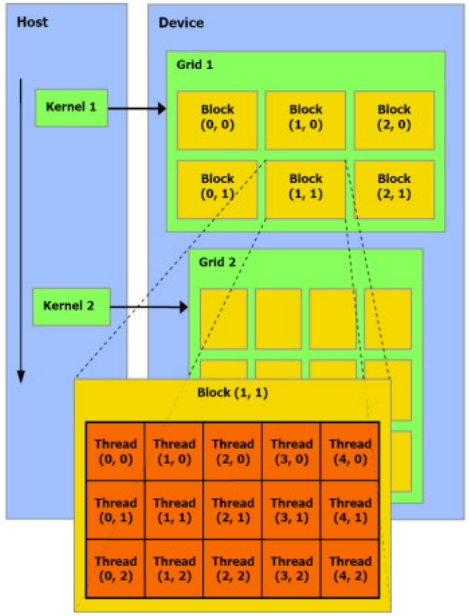
Exemple : GeForce 8800

Architecture GeForce8800

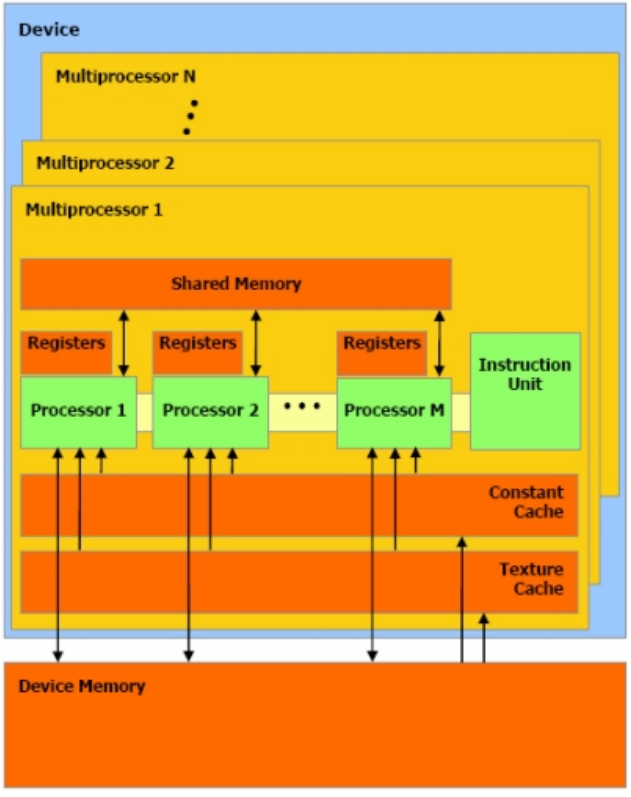


- Architecture de type SIMD (Single Instruction Multiple Datas).
- Peut être vu comme une grosse unité de calcul divisée en multiprocesseurs.
- Chaque multiprocesseurs est composé de plusieurs Stream Processors.

- SIMD $\Longrightarrow$ un même programme (kernel) exécuté plusieurs fois simultanément.
- Chaque exécution = 1 thread (élément de base).
- Threads groupés en warps (32 threads) à l'intérieur de blocs (64 à 512 threads).
- 1 bloc $\Longleftrightarrow$ 1 Multiprocesseur.
- Ces blocs sont réunis dans des grilles.



Architecture GeForce8800



- Niveau GPU : Beaucoup de RAM (> 128 Mo).
- Niveau Multiprocesseur : Mémoire partagée (16 ko), accès très rapide, permet la communication entre threads. Des caches (constante et texture).
- Chaque Stream Processor : des registres (8192 * 32 bits).

CUDA

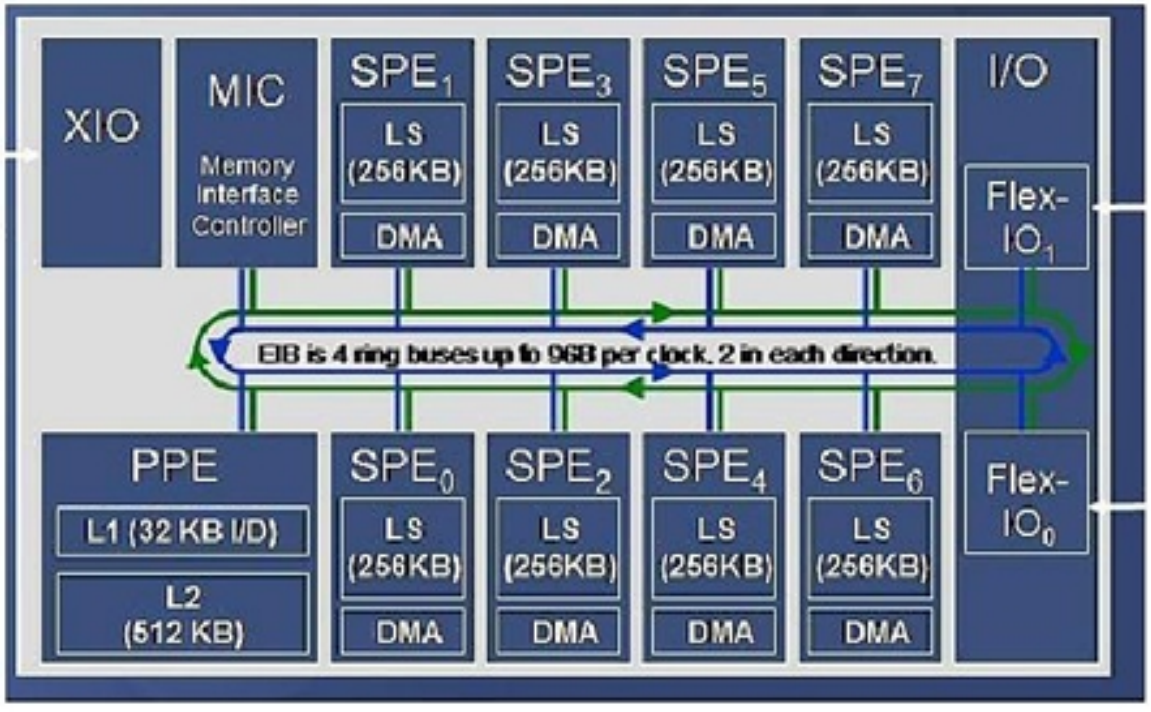
- Ensemble d'extensions au langage C et une API.
- Mot clé principal : *__global__*, placé devant une fonction, indique qu'elle est un kernel, cad une fonction qui va être appelée par le CPU et exécutée par le GPU. Il faut lui indiquer la taille de la grille et la taille de chaque bloc sur lequel le kernel est appliqué.
- *__device__* pour sa part désigne une fonction qui sera exécutée par le GPU mais qui n'est appelable que depuis le GPU.
- L'API CUDA offre quant à elle essentiellement des fonctions de manipulation mémoire en VRAM.

Programmation CUDA

Optimiser un programme consiste donc essentiellement à équilibrer au mieux le nombre de blocs et leur taille : plus de threads par blocs s'avère utile pour mieux masquer la latence des opérations mémoires mais d'un autre côté cela diminue le nombre de registres disponibles par threads.

Outline

Processeurs spécialisés : le CELL



- Développé par Sony, Toshiba et IBM, processeur de la PlayStation 3.
- Un cœur principal (PPE) et 8 cœurs spécifiques (SPE).
- le PPE : processeur PowerPC classique, sans optimisation, « in order », il affecte les tâches aux SPEs.
- les SPEs : constitués d'une mémoire locale (LS) et d'une unité de calcul vectoriel (SPU). Ils ont un accès très rapides à leur LS mais pour accéder à la mémoire principale ils doivent effectuer une requête de transfert asynchrone à un bus d'interconnexion. Les SPEs exécutent les tâches calculatoires.
- Le travail d'optimisation est à la charge du programmeur.

Parallélisme sur le CELL

- les SPU permettent de traiter 4 op 32 bits/cycle (registre de 128 b).
- Programmation explicite des threads indépendante pour chaque cœur.
- Partage explicite de la mémoire : l'utilisateur doit gèrer la copie des données entre cœurs.
  $\Longrightarrow$ Plus difficile à programmer que les GPUs (car pour les GPUs, les threads ne communiquent pas, excepté au début et à la fin).

Processeur CELL : performance crête (registres 128b, SP)
4 (SP SIMD) $\times$ 2 (FMA) $\times$ 8 SPUs $\times$ 3.2 GHz = 204.8 GFlops/socket (en SP)

Outline

Technologie réseau

Technologie	Vendeur	Latence MPI usec, short msg	Bande passante par lien unidirectionnel, MB/s
NUMAlink 4	SGI	1	3200
QsNet II	Quadrics	1.2	900
Infiniband	Mellanox	1.07	2500 Double speed DDR Quad speed QDR
High Performance Switch	IBM	5	1000
Myri-10G	Myricom	2.6	2500
Ethernet 10 Gb		< 10	2600
Ethernet 1 Gb		> 40	50 à 80

Outline

Conclusions

- Vers une augmentation du nombre de cœurs plutôt que de la fréquence d'horloge : problème de finesse de gravure, d'échauffement ...
- Programmation de plus en plus complexe :
  - Nécessité d'une bonne connaissance des architectures.
  - Pas de croissance des performances au niveau du cœur $\Longrightarrow$ parallélisation obligatoire pour un gain de performance.
  - Nécessité d'élaborer des algorithmes et des programmes capables d'exploiter ce grand nombre de processeurs.
  - Programmation sur des architectures hybrides avec différents types de processeurs.
- Exploitation des performances des processeurs graphiques : nécessité de produire un code extrêmement parallélisé, capable d'utiliser TOUS les "threads", sans quoi, on n'atteint que quelques pour cents de l'utilisation de la puissance disponible.