Formation en Calcul Scientifique - LEM2I

Architecture des calculateurs

Violaine Louvet [1]

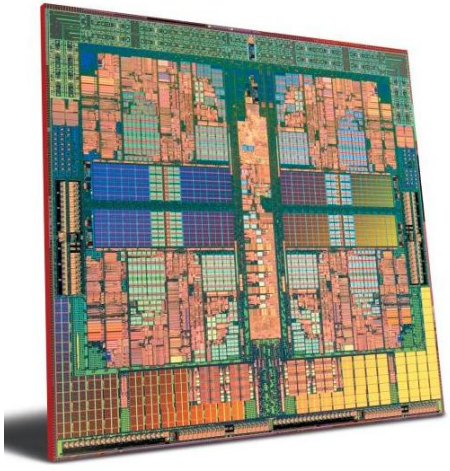
[1] Institut Camille jordan - CNRS

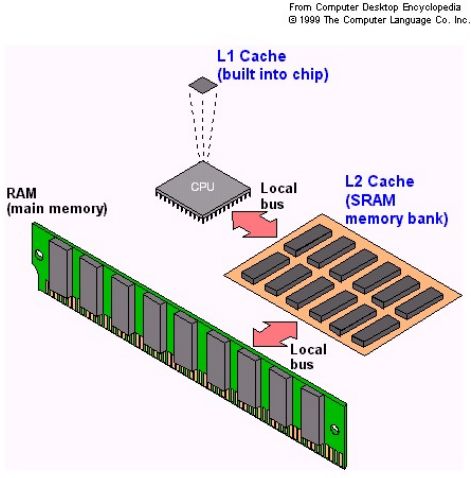
12-13/09/2011

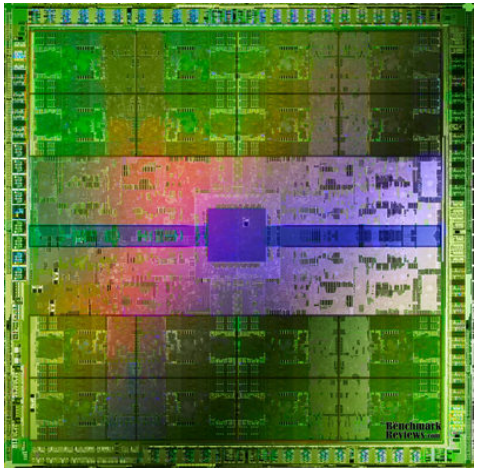
Introduction

Décoder la relation entre l'architecture et les applications :

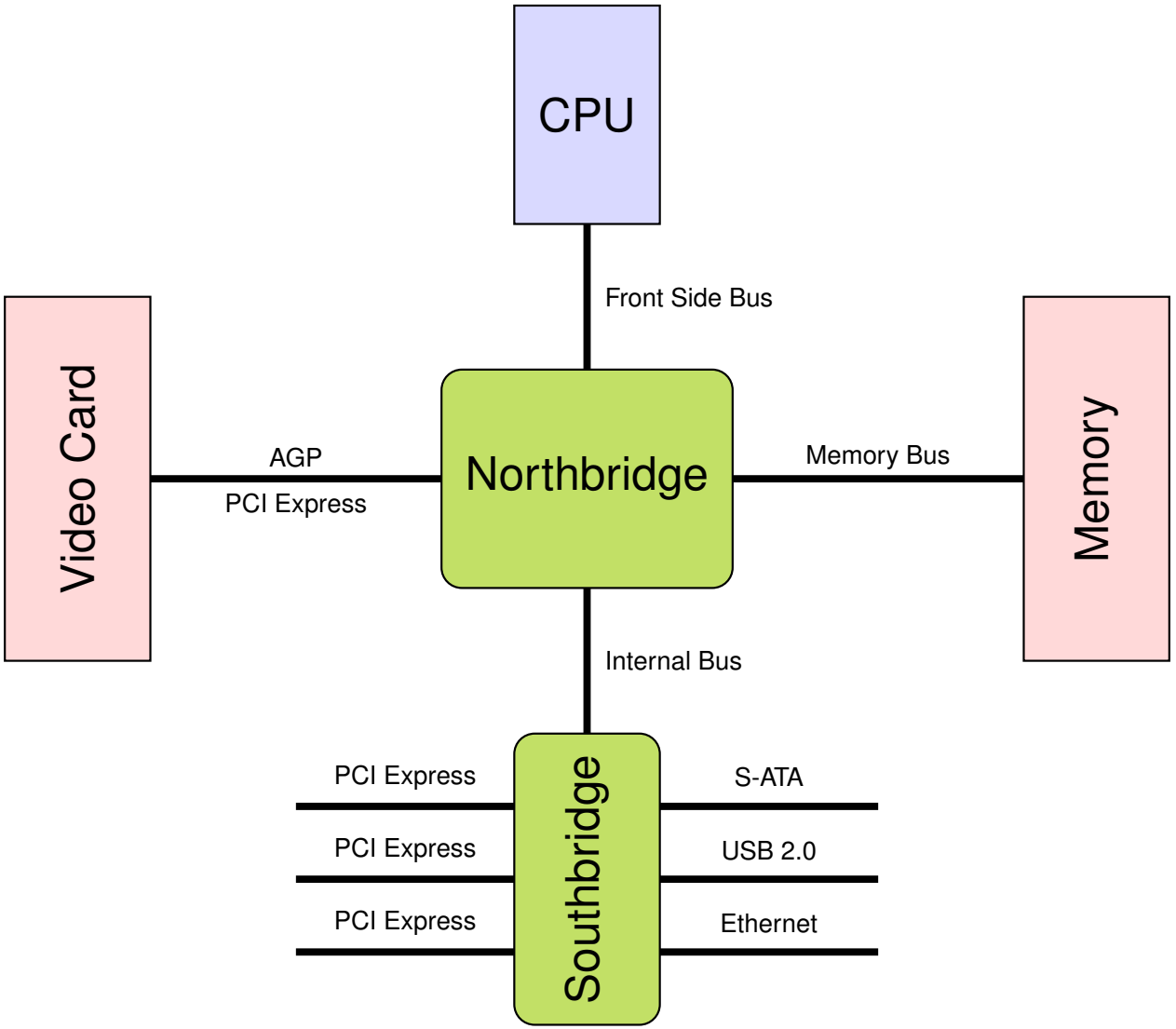
- Choisir son architecture de calcul en fonction de ses besoins
- Adapter les méthodes numériques, les algorithmes et la programmation
- Comprendre le comportement d'un programme
- Optimiser les codes de calcul en fonction de l'architecture





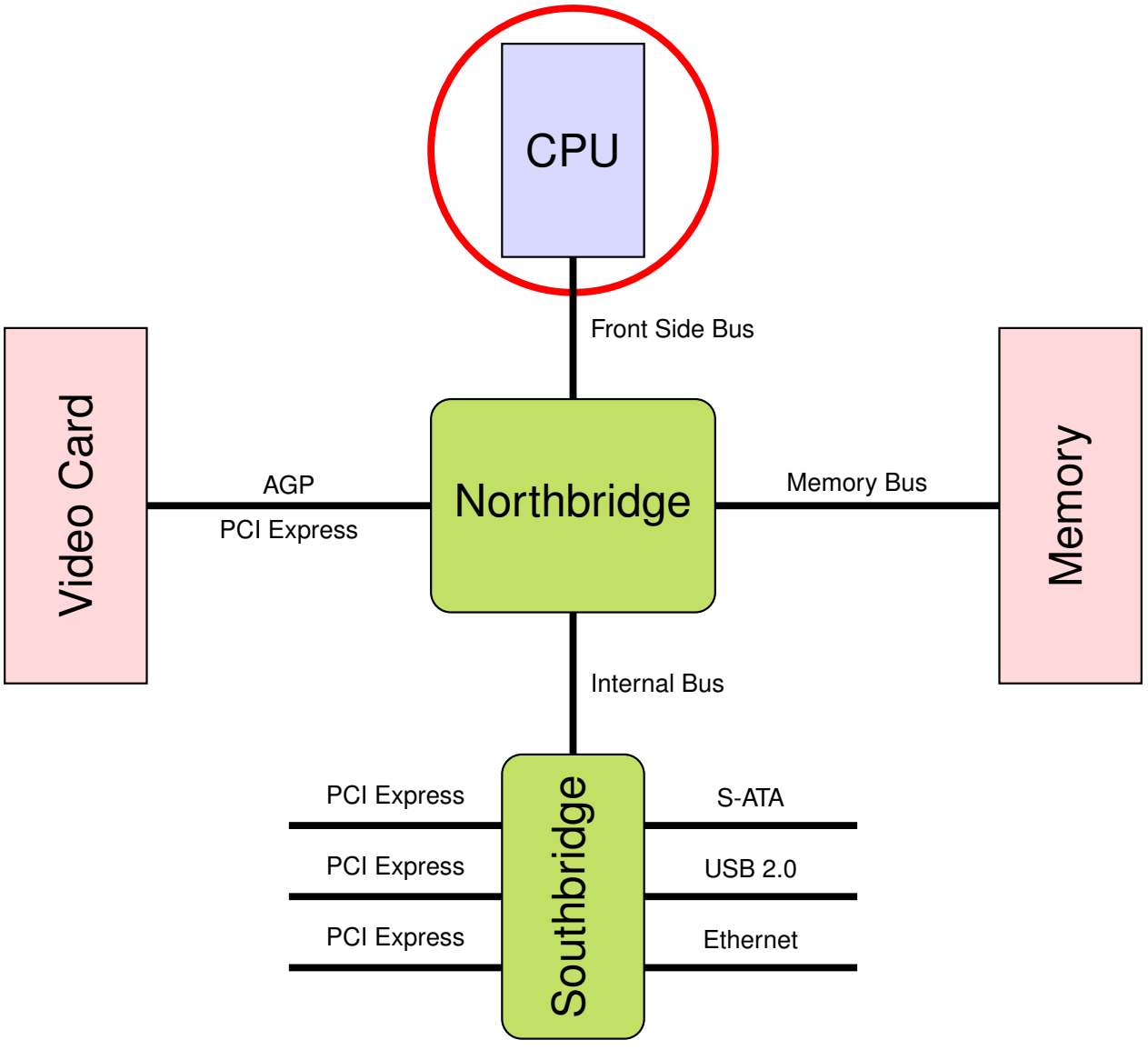


Architecture générale



Sommaire

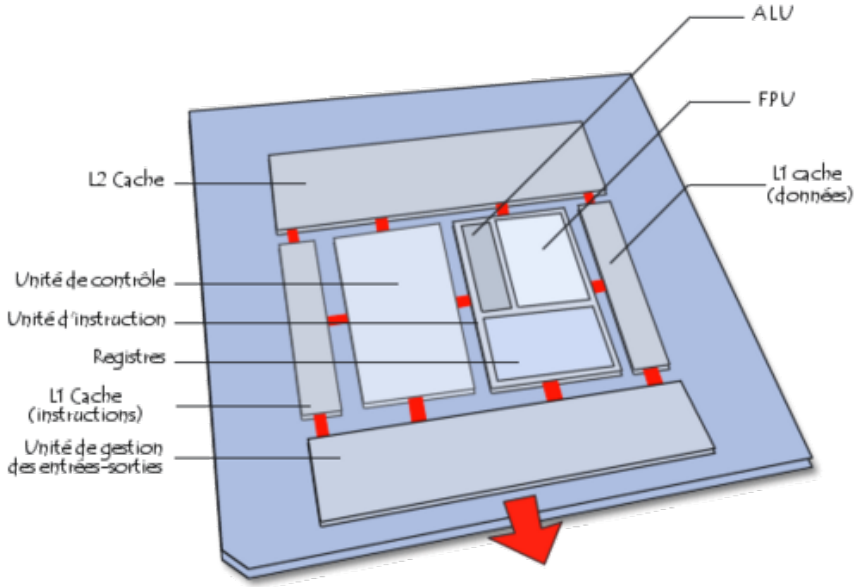
Le processeur



Composition d'un processeur

Eléments importants du processeur :

- Plusieurs Unité Arithmétique et Logique (UAL), qui prennent en charge notamment les calculs arithmétiques élémentaires ce qui permet de traiter plusieurs instructions en même temps
- Registres : mémoires de petite taille (quelques octets) rapides
- Unité de calcul en virgule flottante (en anglais Floating Point Unit - FPU), qui permet d'accélérer les calculs sur des nombres flottants
- Mémoires caches, qui diminuent les temps d'accès à la mémoire



Fonctionnement

- Le processeur est rythmé par une horloge défini par sa fréquence (MHz ou GHz)
- A chaque impulsion d'horloge (exemple classique d'un processeur à 5 étages) :
  1. lecture de l'instruction (IF, Instruction Fetch)
  2. décodage de l'instruction (ID, Instruction Decode)
  3. exécution de l'instruction (EX, Execute)
  4. écriture ou chargement depuis la mémoire en fonction du type de l'instruction (MEM, Memory)
  5. stockage du résultat dans un registre (WB, Write Back)
- Plus la fréquence est élevée et plus le processeur peut traiter les instructions rapidement

5 cycles sont nécessaires pour accomplir une instruction

Jeux d'instructions (ISA, Instructions Set Architecture)

- Ensemble des opérations élémentaires qu'un processeur peut accomplir
- Plusieurs familles de processeurs possédant chacune un jeu d'instructions propre :
  - 80x86 : le « x »représente la famille. On parle ainsi de 386, 486, 586, 686, etc.
  - ARM
  - IA-64
  - PowerPC ...

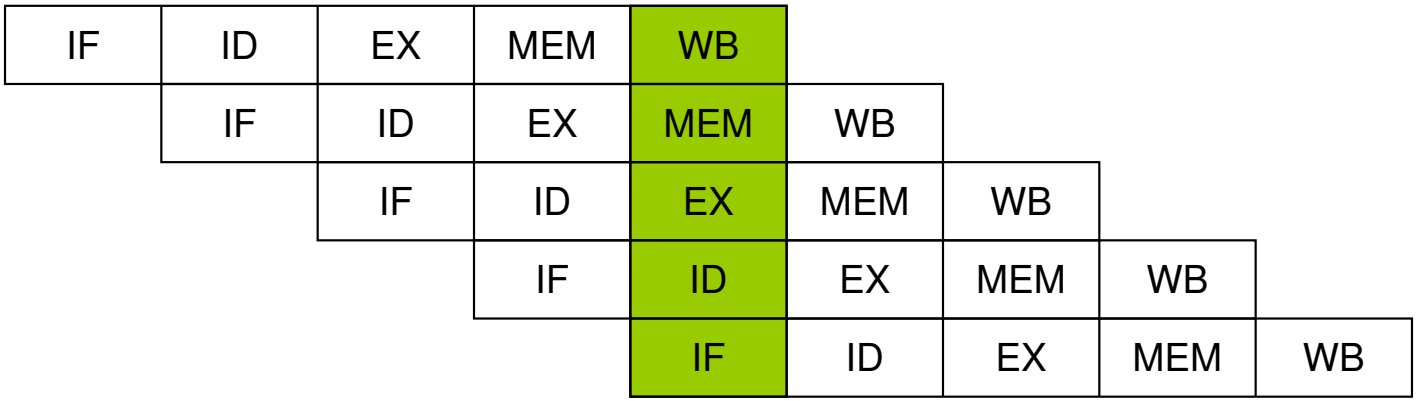
Attention

Un programme réalisé pour un type de processeur ne peut pas fonctionner directement sur un système possédant un autre type de processeur.

Pipeline

Pipeline (ou pipelining) : technologie visant à permettre une plus grande vitesse d'exécution des instructions en parallélisant des étapes.

- L'ordre des étapes est invariable
- Chaque étape est réalisée par un circuit différent



5 instructions en parallèle en 9 cycles

Parallélisation de données

SIMD (Single Instruction Multiple Data) : appliquer la même instruction simultanément à plusieurs données

- Jeux d'instructions :
  - SSE (actuellement SSE4) chez Intel
  - 3DNow ! chez AMD
- Processeur vectoriel : effectue des opérations sur des vecteurs. Les données chargées dans le pipeline ne sont plus des scalaires mais des vecteurs. Exemple : NEC.

Exemple SSE

Algorithme :

```
for i = 1,n
   x[i] = sqrt(x[i])
```

Travail du processeur :

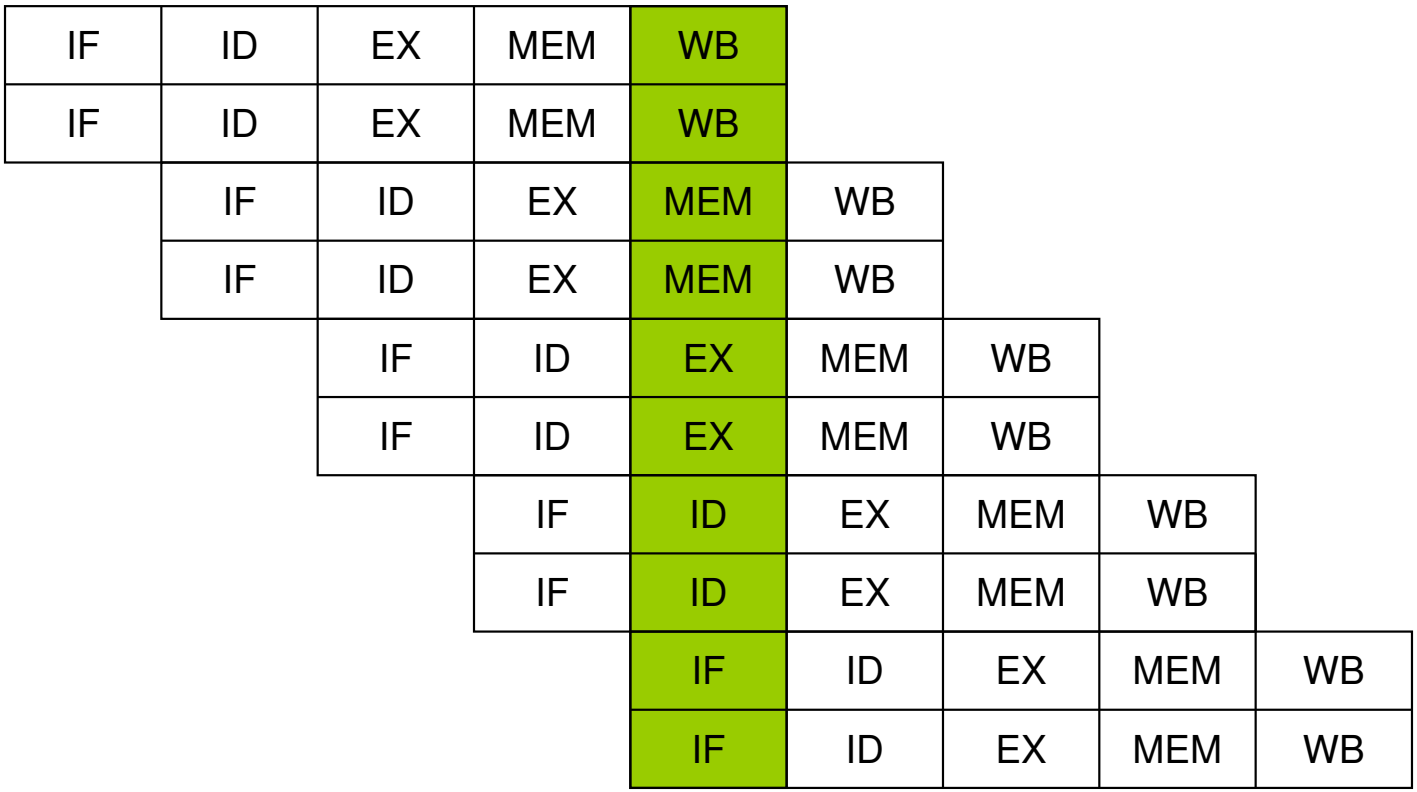
```
for i = 1,n
    load x[i] to the floating-point register
    calculate the square root
    write the result from the register to memory
```

Avec SSE :

```
for {i1,i2,i3,i4} in {1:n}
    load x[i1],x[i2],x[i3],x[i4] to the SSE register
    calculate 4 square roots in one operation
    write the result from the register to memory
```

Architectures superscalaires

Exécution de plusieurs instructions simultanément, chacune dans un pipeline différent. Le processeur dispose alors de plusieurs unités de calcul.



Hyperthreading

- SMT = Simultaneous MultiThreading, plus connu sous le nom d'Hyperthreading chez Intel.
- Principe : créer deux processeurs logiques sur une seule puce, chacun doté de ses propres registres. Ces deux unités partagent les éléments du cœur physique comme le cache et le bus système.
- Dégradation des performances individuelles des threads mais amélioration des performances de l'ensemble.

Attention

Ce fonctionnement est généralement activé par défaut.

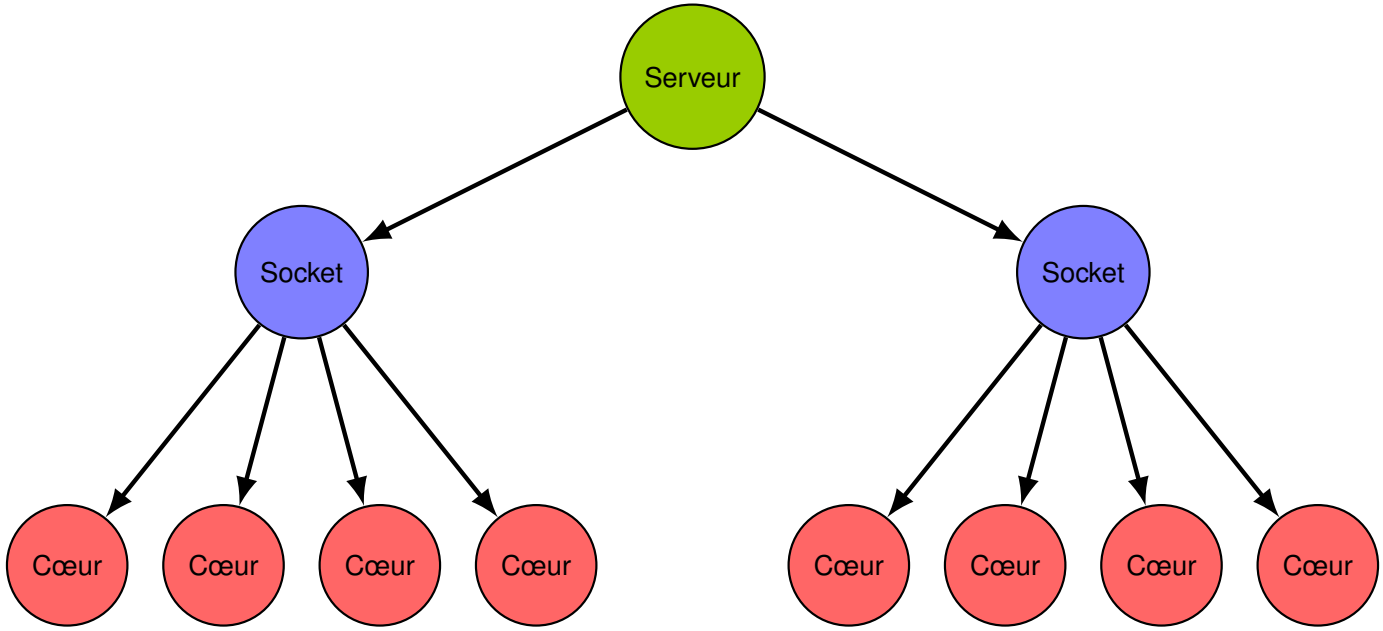
En pratique

Sur un core I7, commande en ligne *cpuinfo* :

```
Intel(R) Core(TM) i7  Processor (Intel64 Q 820  )
=====  Processor composition  =====
Processors(CPUs)  : 8
Packages(sockets) : 1
Cores per package : 4
Threads per core  : 2
```

Multi-cœurs, socket

On évite de parler de CPUs : il faut distinguer le support (socket) de l'unité de calcul elle-même (cœur).
Le socket ou slot est le connecteur qui interfère entre la carte mère d'un ordinateur et le processeur lui-même.



Serveur bi-sockets quadri-cœurs

Flops et performance du processeur

- Opérations (additions ou multiplications) à virgule flottante par seconde = FLoating point Operations Per Second
- Puissance crête (point de vue théorique) : mesure les performance des unités de calcul en virgule flottante (FPU) contenues dans le cœur.
  - Processeur Intel Harpertown quad core, 3 GHz : 4 opérations flottantes possibles par cycle, 4 cœurs par nœud : $4 \times 4 \times 3 = 48 GFlops$.
  - Processeur PowerPC 450 (Blue Gene/P), 850 MHz : 4 opérations flottantes possibles par cycle, 4 cœurs par nœud : $4 \times 4 \times 0.850 = 13.6 GFlops$.
- D'un point de vue pratique, la puissance d'une machine dépend de l'ensemble de ses composants : fréquence du processeur, accès mémoire, vitesse des bus, complexité de l'architecture ... mais aussi charge de la machine, système d'exploitation ... On est souvent loin de la puissance théorique ...

Performances théoriques et réelles

Exemple : résolution du système linéaire dense $n \times n$ de type $Ax = b$ par élimination de Gauss. La complexité de l'algorithme est de l'ordre de $\frac{1}{3}n^3$ opérations flottantes.

Pour un calculateur de 48 GFlops = 48×10^9 Flops :

n	Nombre d'opérations	Temps
1000	3.3×10^8	0.0068 s
10 000	3.3×10^{11}	6.875 s
100 000	3.3×10^{14}	6875 s = 1h 44 mn 35 s
1 000 000	3.3×10^{17}	79j

Mais ...

L'architecture intrinsèque du processeur peut impacter négativement la performance :

- Partage de certaines ressources
- Gestion des dépendances entre instructions
- Prédiction de branchement ...

Pourquoi les multi-cœurs ?

Quelques ordres de grandeurs

	Single core	Dual core	Quad core
Aire du cœur	A	$\sim A/2$	$\sim A/4$
Puissance du cœur	W	$\sim W/2$	$\sim W/4$
Performance du cœur	P	$0.9 \times P$	$0.8 \times P$
Performance du processeur	P	$1.8 \times P$	$3.2 \times P$

Exemple

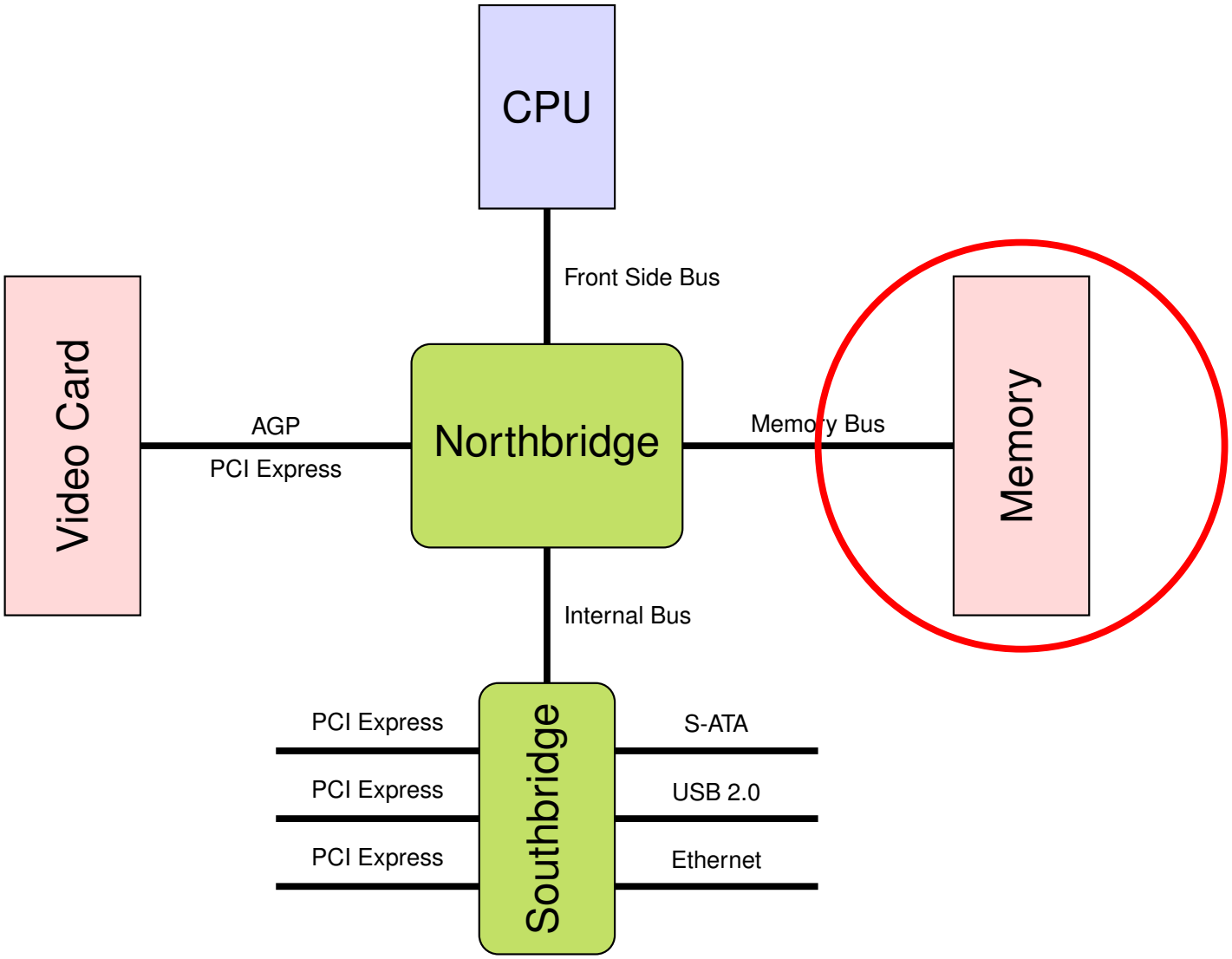
- Un serveur bi-processeur simple cœur cadencé à 3.6 GHz

  a une puissance équivalente à
- Un serveur bi-processeur quadri-cœurs cadencés à 1.8 GHz

  mais celui-ci consomme 2 fois moins (rappel : $P \sim f^3$).

Sommaire

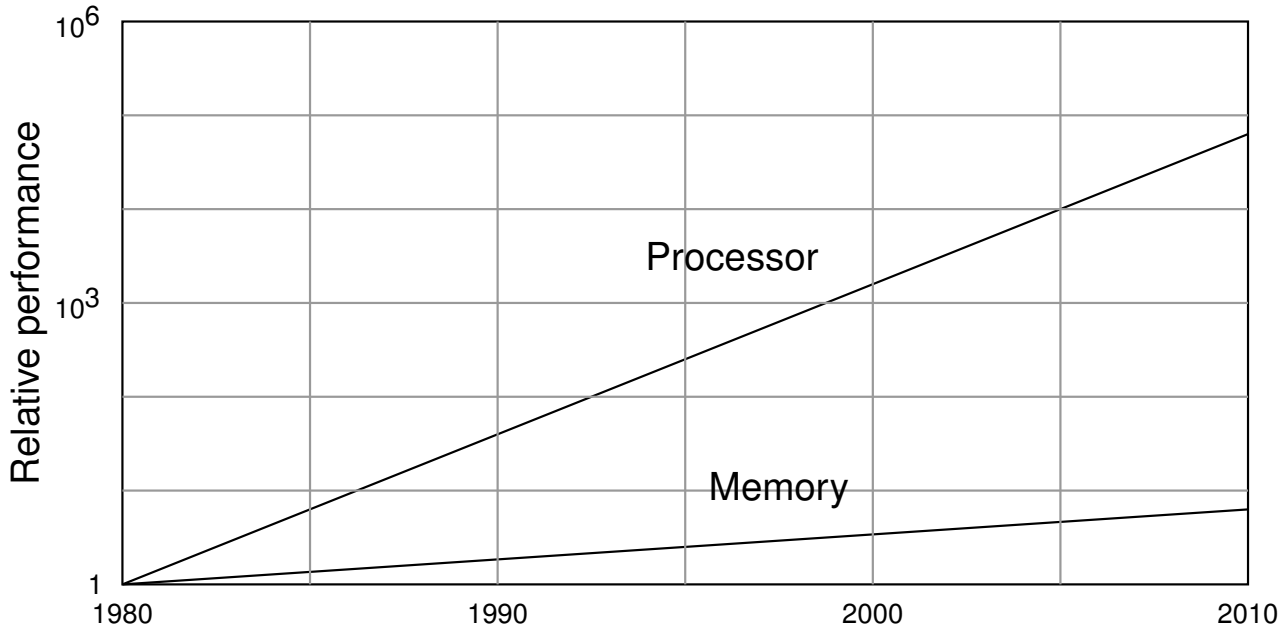
La mémoire


CPU
Front Side Bus
Video Card
AGP
PCI Express
Northbridge
Memory Bus
Memory
Internal Bus
PCI Express
PCI Express
PCI Express
Southbridge
S-ATA
USB 2.0
Ethernet

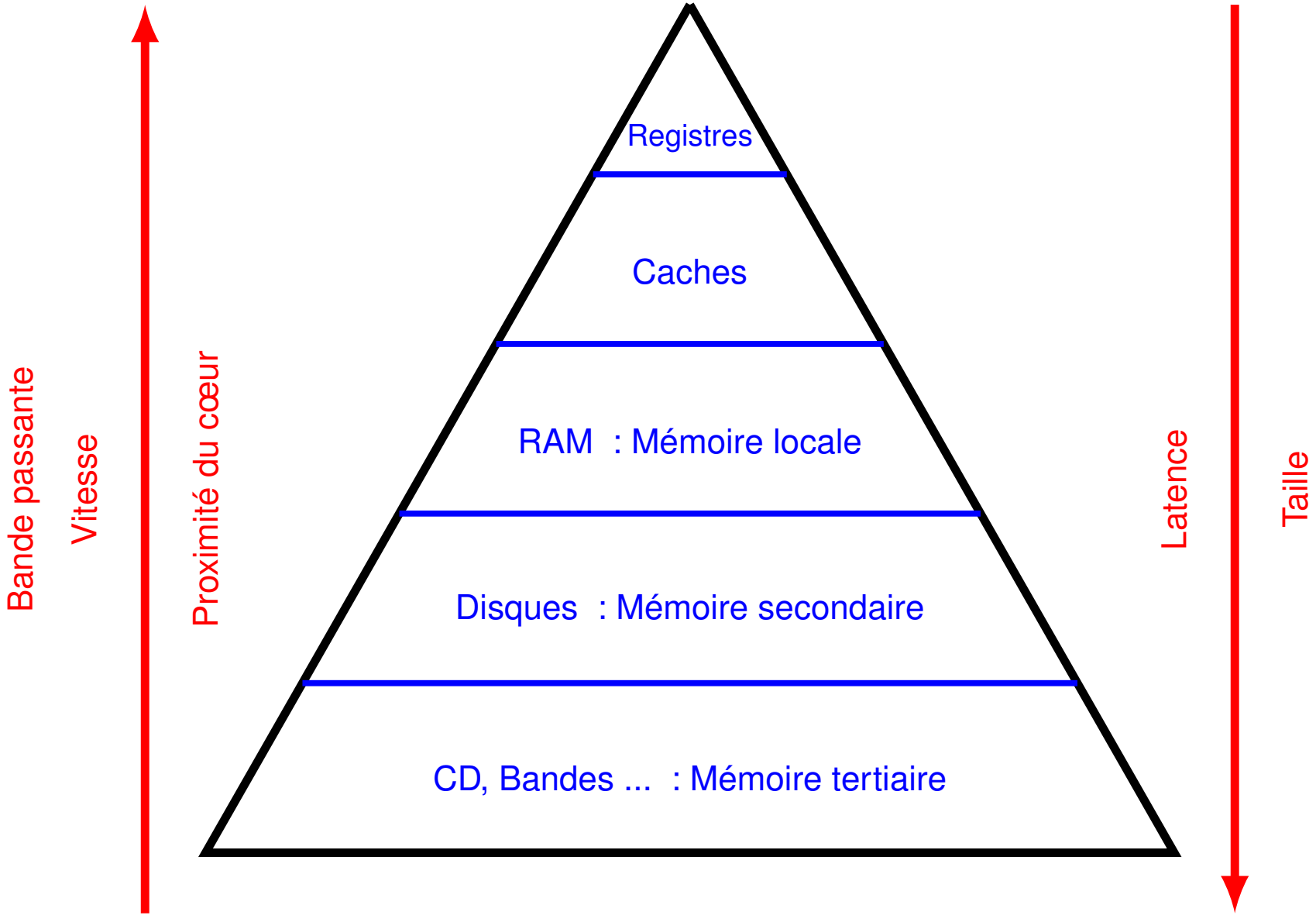
Problématique

« Memory Wall »

$$Vitesse_{CPU} \gg Vitesse_{Memory}$$



Hiérarchie Mémoire



Technologies mémoire

DRAM (Dynamic Random Access Memory)

- Avantages :
  - Coût
  - Densité (donc capacité)
- Désavantages :
  - Latence/Débit
  - Contraintes d'utilisation (besoin d'un rafraîchissement périodique)

SRAM (Static Random Access Memory)

- Avantages :
  - Rapidité (cycle SRAM 8 à 16 fois plus rapide qu'un cycle DRAM)
  - Stabilité (pas de rafraîchissement)
- Désavantages :
  - Coût (8 à 16 fois plus élevé que la mémoire DRAM)
  - Densité

Agencement et caractéristiques

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.

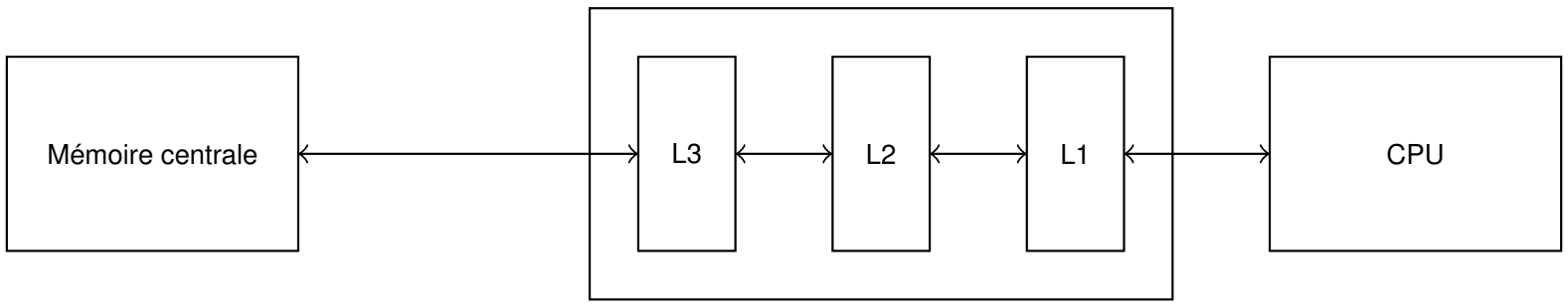
Type	Taille	Vitesse	Coût/bit
Registre	< 1 KB	< 1 ns	$$$$
SRAM On-chip	8 KB - 6 MB	< 10 ns	$$$
SRAM Off-chip	1 MB - 16 MB	< 20 ns	$$
DRAM	64 MB - 1 TB	< 100 ns	$
Flash	64 MB - 32 GB	< 100 μs	c
Disk	40 GB - 1 PB	< 20 ms	$\sim$0

Accès mémoire : problématique

Quels facteurs influencent les performances des accès mémoire ?

- Localisation : cache, RAM, disque ?
- Manière dont elles sont accédées :
  - au travers d'un chipset (NorthBridge)
  - directement par le processeur
  - via le processeur voisin ...

Plus une donnée est proche du processeur, plus elle est accédée rapidement.



Principes de localité

Localité spatiale

Lorsqu'un programme accède à une donnée ou à une instruction, il est probable qu'il accédera ensuite aux données ou instructions voisines

Localité temporelle

Lorsqu'un programme accède à une donnée ou à une instruction, il est probable qu'il y accédera à nouveau dans un futur proche

Exemple

```
subroutine sumVec(vec,n)
    integer :: n
    integer :: vec(n)
    integer :: i,sum=0
    do i = 1,n
        sum = sum + vec(i)
    end do
end subroutine
```

- Bonne localité spatiale des données du tableau `vec` : accès en séquence
- Bonne localité temporelle de la donnée `sum` : accès fréquent
- Bonne localité spatiale et temporelle des instructions : accès en séquence et fréquemment

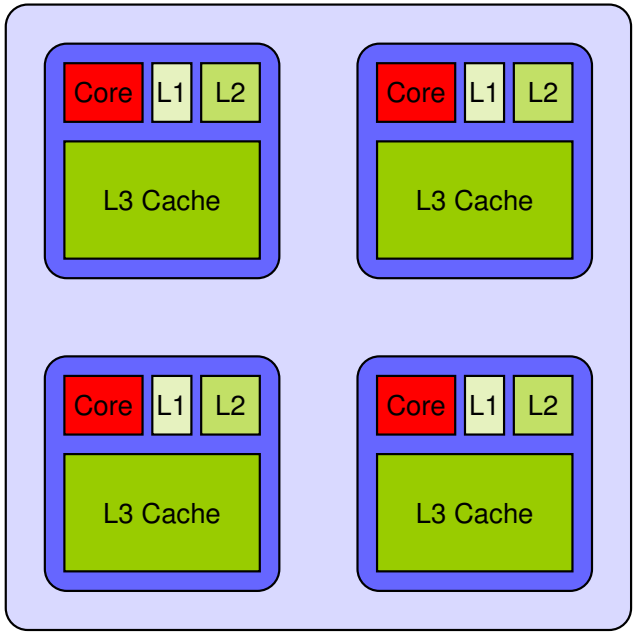
Fonctionnement des caches

- Le cache est divisé en lignes (ou blocs) de mots
- 2 niveaux de granularité :
  - le CPU travaille sur des mots (par ex 32 ou 64 bits)
  - les transferts mémoire se font par ligne (ou bloc, par ex 256 octets)
- Les lignes de caches sont organisées en ensembles à l'intérieur du cache, la taille de ces ensembles est constante et appelée le degré d'associativité.
- Exploitation de la localité spatiale : le cache contient des copies des mots par lignes de cache
- Exploitation de la localité temporelle : choix judicieux des lignes de cache à retirer lorsqu'il faut rajouter une ligne à un cache déjà plein

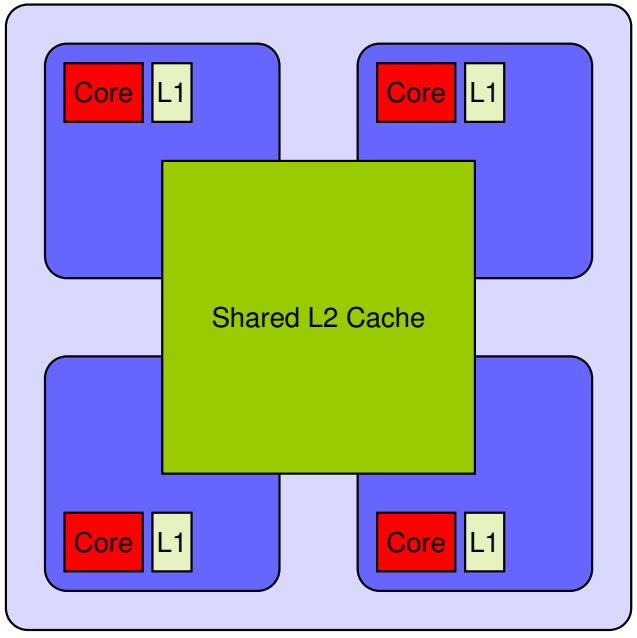
Lorsque le processeur tente d'accéder à une information (instruction ou donnée)

Si l'information se trouve dans le cache (hit), le processeur y accède sans état d'attente, sinon (miss) le cache est chargé avec un bloc d'informations de la mémoire

Organisation des caches



Caches indépendants



Caches partagés

- Problème de cohérence
- Problème d'accès concurrent

Mémoire virtuelle, mémoire physique

- Accès cache : $\sim$ 15 cycles
- Accès mémoire : $\sim$ 200 cycles

Accès à 100 éléments 100 fois

- sans cache : 2 000 000 cycles
- avec cache : 168 500 cycles

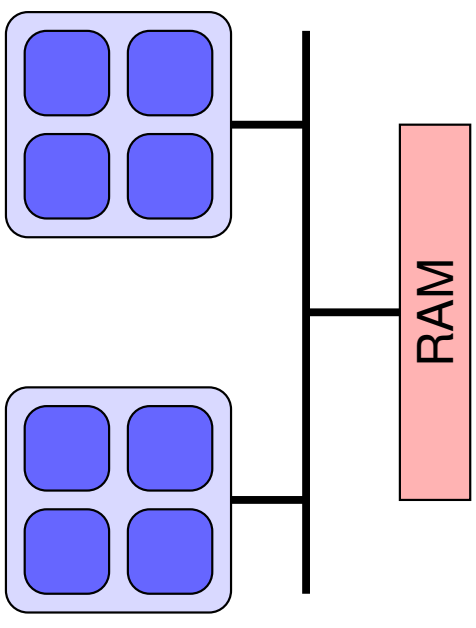
$>$ 91% de gain.

- Mémoire physique : mémoire effectivement présente sur l'ordinateur
- Mémoire logique : mémoire virtuelle. Nécessite une fonction de correspondance vers la mémoire physique
- Chaque processus dispose de son propre espace d'adressage virtuel
- Cet espace virtuel est en correspondance avec l'espace d'adressage physique
- Ils sont divisés en pages
- Taille des pages : de 4 KB à 4 MB selon les architectures

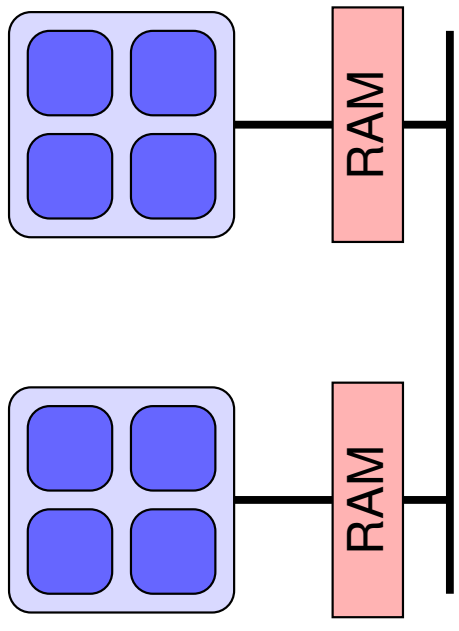
UMA/NUMA

- Modèle SMP : relie plusieurs processeurs à une seule mémoire centralisée. Architecture UMA (Uniform Memory Access) : le temps nécessaire pour accéder à un emplacement quelconque en mémoire est le même pour tous les processeurs.
- Modèle NUMA (Non Uniform Memory Access) : les processeurs sont capables d'accéder à la totalité de la mémoire du système mais les temps d'accès mémoire sont non uniformes en fonction de la distance mémoire/cœur.

Architecture SMP



Architecture NUMA



Page faults / Caches miss

- Défaut de cache (cache miss) : lorsque le processeur essaie d'accéder une donnée qui n'est pas dans le cache
  - Raté obligatoire (compulsory miss) : premier accès au bloc pendant l'exécution du programme. Inévitable.
  - Raté par capacité insuffisante (capacity miss) : accès à un bloc qui a été remplacé par un autre dans le cache car la capacité du cache est insuffisante.
- Défaut de page (page fault) : lorsqu'un processus tente d'accéder à une adresse logique correspondante à une page non résidente en mémoire centrale $\longrightarrow$ interruption du processus le temps de charger la page.
  - Temps de traitement d'un défaut de page : $\sim 25ms$

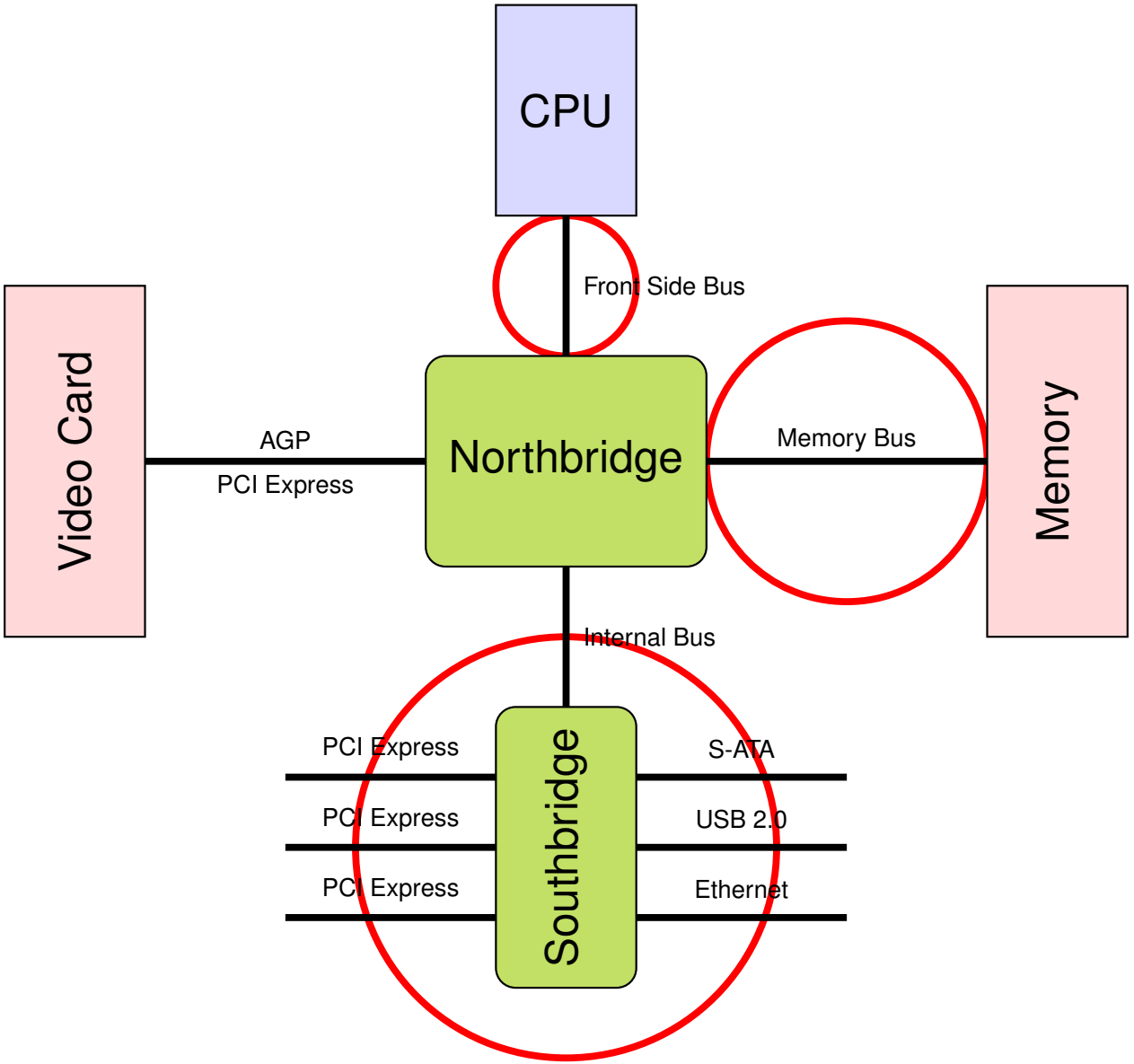
Exemple : Balayage d'un grand tableau

```
do  i=1,n
    do  j=1,m
        ...  a(i,j)  ...
    end  do
end  do
```

Hypothèse : architecture 32 bits, cache L1 de 16 Ko avec des lignes de 64 octets. Quel est le taux d'échec si $n = m = 256$? Taux d'échec = nombre total d'échecs divisé par le nombre total d'accès.

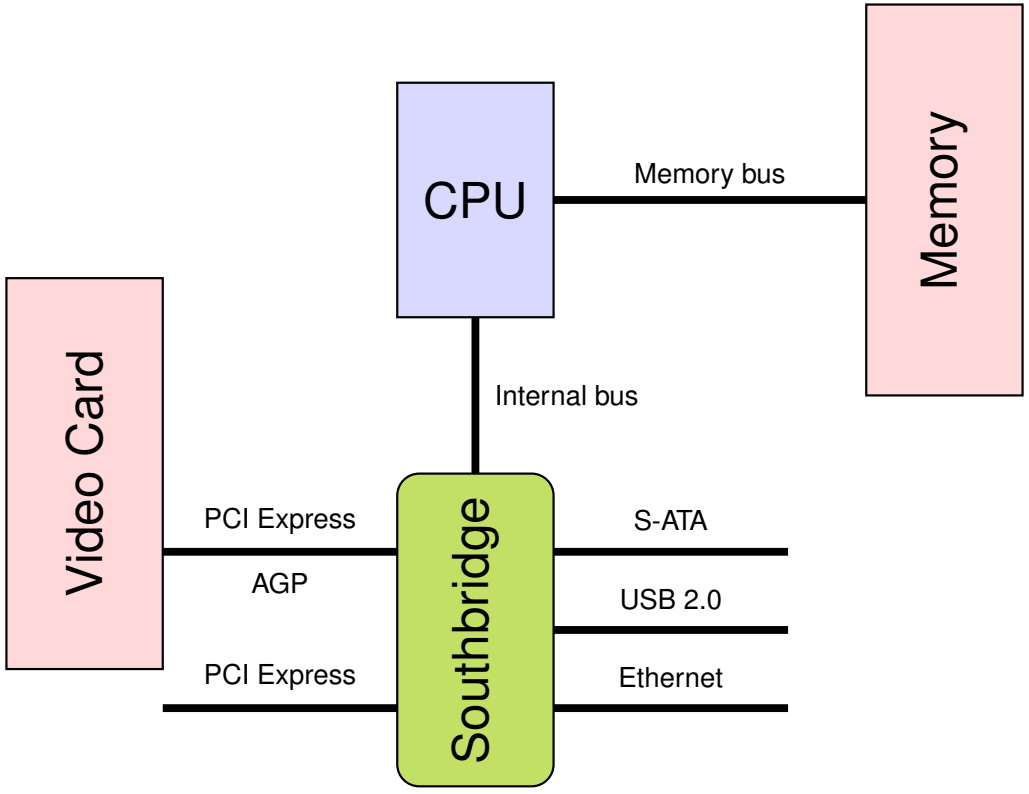
Sommaire

Communications



Architecture générale sans Northbridge

Technologies HyperTransport d'AMD (Opteron ...) ou QuickPath Interconnect d'Intel (Nehalem, Sandy Bridge ...)



Les différents types de communications

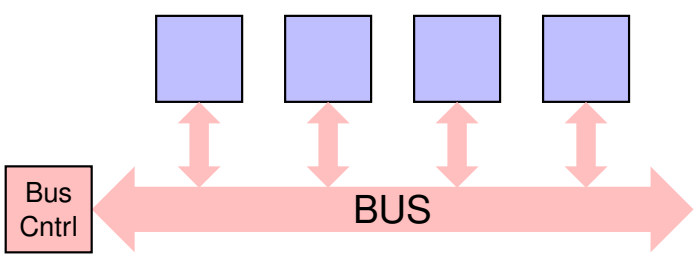
Facteurs importants :

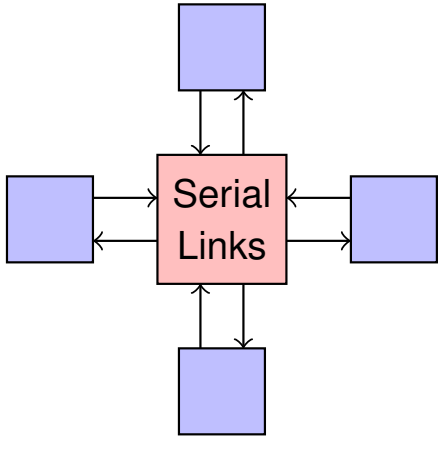
- la fréquence (en Hz)
- la largeur du bus : nombre de bits d'informations pouvant être transmis

Bus → point à point

La plupart des systèmes de communications ont évolués ces dernières années des bus vers des liens séries :

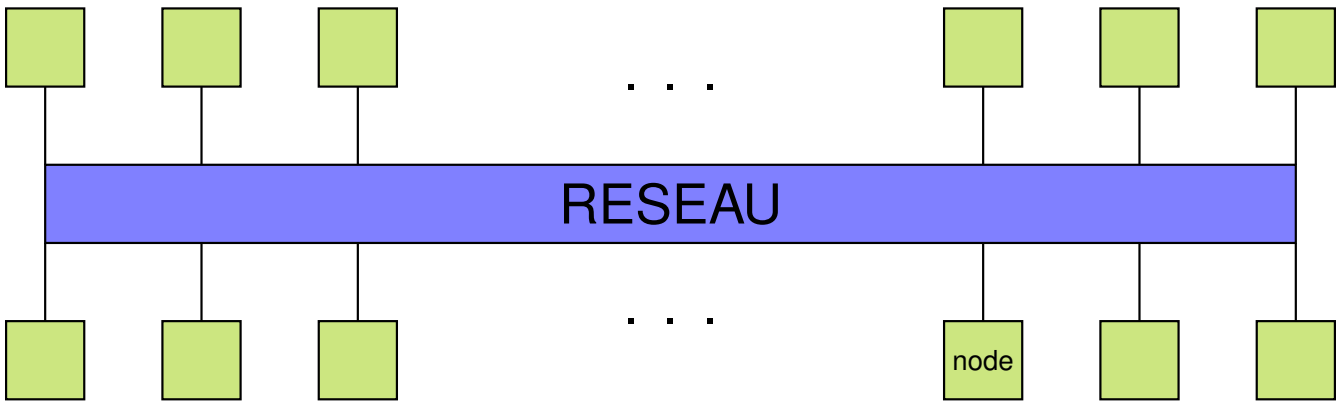
- Communication mémoire-CPU : du Front Side Bus vers les technologies HyperTransport (HT) ou QuickPath Interconnect (QPI)
- Communication CPU-GPU... : PCI/PCI-X vers PCI Express (PCIe)
- Communication CPU-Disque : IDE vers SATA, SCSI vers SAS





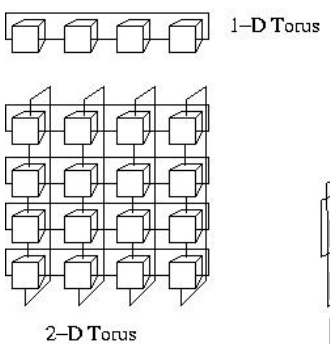
Sommaire

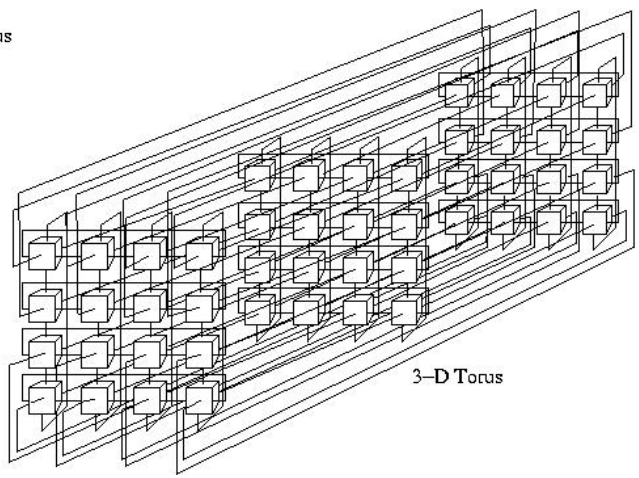
Réseaux

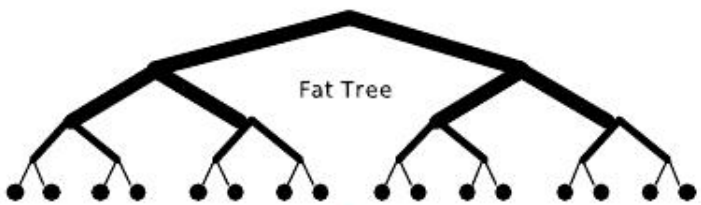


Caractéristiques des réseaux

- La bande passante (bandwidth) est le débit maximal.
  - La bande passante effective est généralement inférieure à la bande passante physique (souvent à cause de l'overhead du protocole)
  - Ce qui intéresse l'utilisateur est la bande passante MPI
  - Les performances bas niveaux sont intéressantes pour le stockage
- La latence : temps d'initialisation de la communication.
- La distance maximum entre deux nœuds.
- La topologie : tore, fat-tree ...

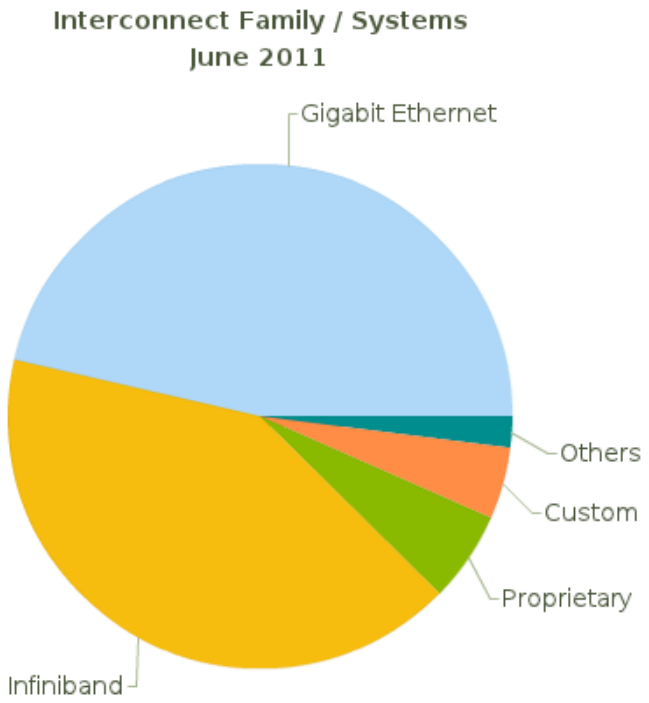






Technologies réseaux

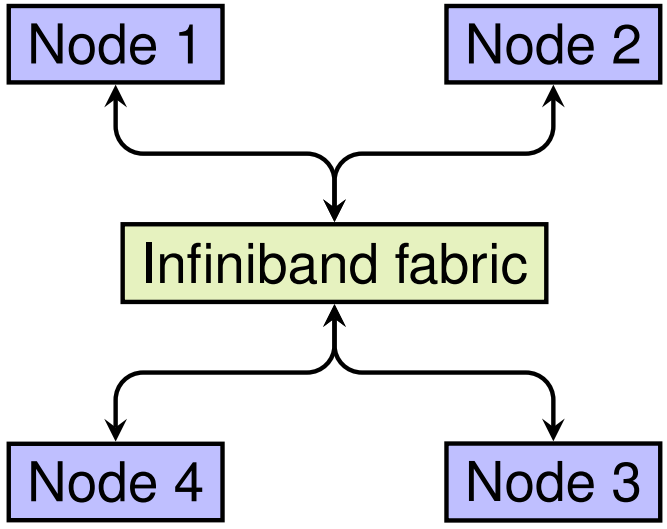
Technologie	Latence	Bande passante
1 Gb Ethernet	-	1 Gb/sec
10 Gb Ethernet	$\sim$10 μs	10 Gb/sec
Infiniband	$<$ 2 μs	10, 20 & 40 Gb/sec
Myrinet	2.5 - 5.5 μs	10 Gb/sec



Interconnect Family / Performance
June 2011

Réseau infiniband

Réseau haute performance très flexible : liaison entre serveurs de données, entre baies de stockage ou interconnexion des nœuds d'un cluster de calcul



- Le fabric peut être un switch ou un ensemble de plusieurs switchs interconnectés et de routeurs
- Le node peut être un serveur, un périphérique d'entrées/sorties...
- Chaque connexion est une connexion point à point, la capacité du réseau est entièrement dédiée de bout en bout

Sommaire

Stockage

Problématique complexe :

- Hardware : technologies disques (SSD, SATA, SAS, ...)
- Systèmes de fichiers :
  - local (ext3, ntfs ...)
  - partagés, parallèles (nfs, Lustre, ...)
- Interconnexion (directe : DAS, par le réseau : SAN, NAS)
  - Liaison directe : SATA, SCSI, ...
  - technologie réseau : Fibre Channel, iSCSi...

Caractéristiques à prendre en compte :

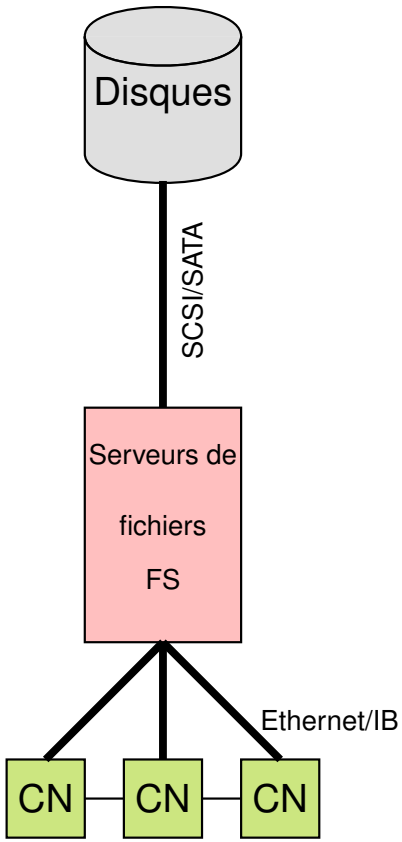
- Volumétrie (et évolutivité)
- Performance : débits ...
- Accès aux données (FS parallèle ...)
- Sécurité (stockage court terme, long terme, haute disponibilité)
- Densité, consommation électrique
- Le coût !!

IOPS

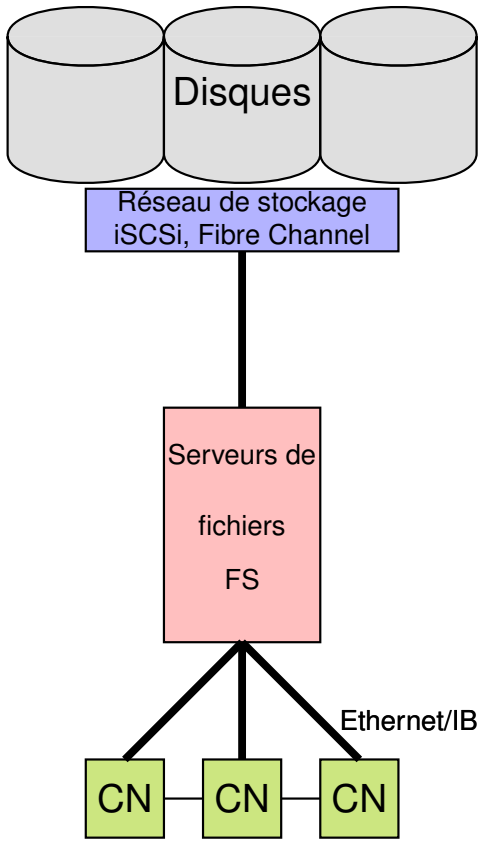
Nombre d'I/O pouvant être traitées par secondes

Principales solutions de stockage

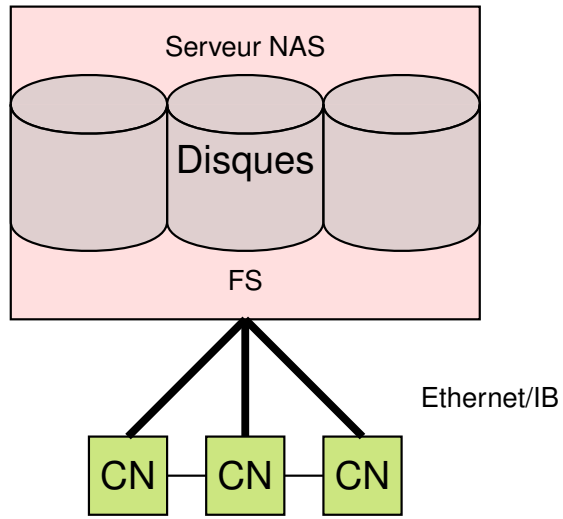
DAS
Direct Attached Storage



SAN
Storage Area Network



NAS
Network Attached Storage



Goulot d'étranglement

Le débit global est dépendant de toute la chaîne d'I/Os (baie, serveur, interconnexion ...).

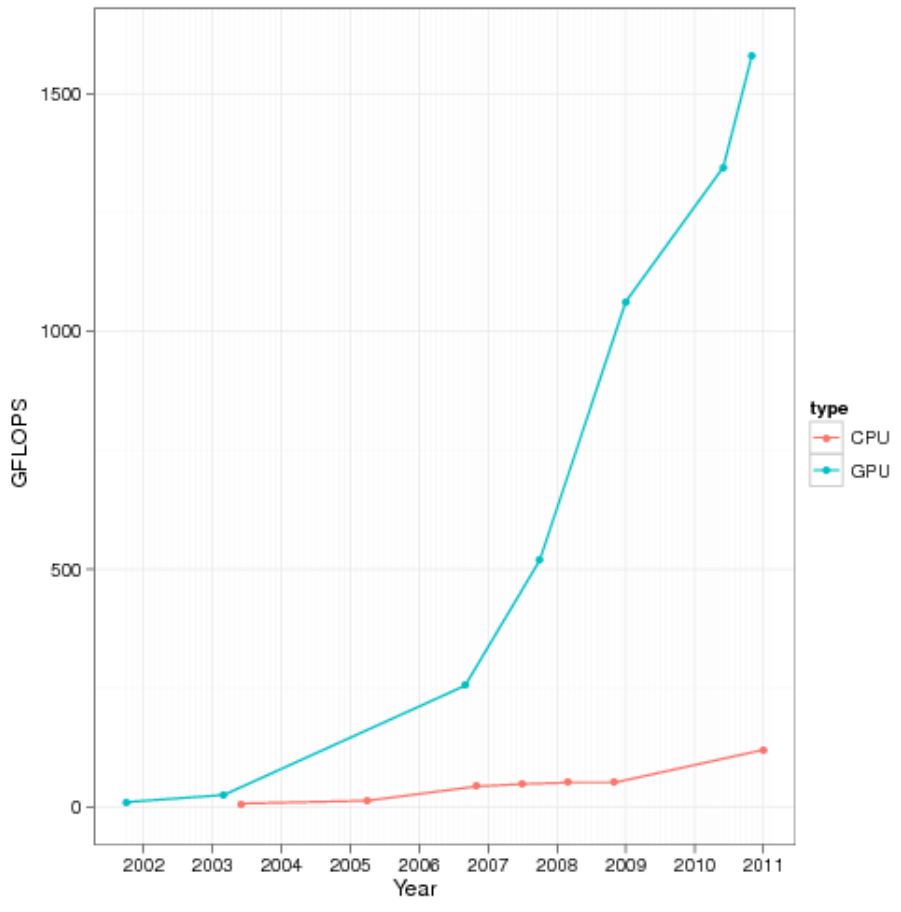
Bottleneck

Loi d'Amdahl sur l'équilibre des machines de calcul : une instruction par seconde requiert un octet de mémoire et un bit/seconde de capacité d'entrée-sortie.

- La capacité de stockage augmente beaucoup plus vite que la vitesse d'accès
- Le débit vers les disques augmente moins vite que la puissance de calcul
- La quantité de données générées augmente avec la puissance de calcul
- L'approche classique un fichier/processus génère de plus en plus de fichiers
- Trop de fichiers = saturation des systèmes de fichiers (nombre maximum de fichiers et espace gaspillé à cause des tailles de bloc fixes)
- Le temps passé dans les I/O croît (surtout pour les applications massivement parallèles)

Sommaire

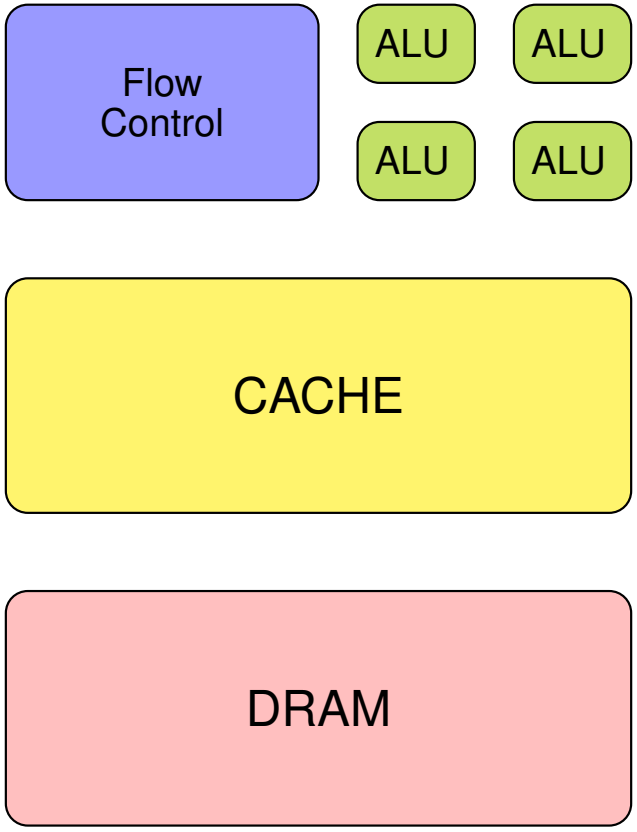
Pourquoi les GPU ?



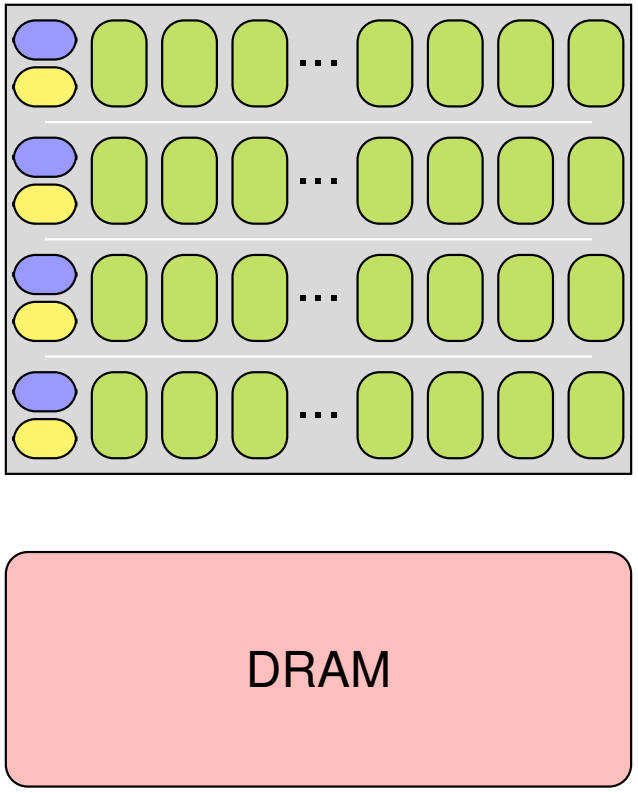
A performance théorique égale, les plateformes à base de GPU :

- occupent moins de place
- sont moins chères
- sont moins consommatrices d'électricité

Architecture comparée d'un CPU et d'un GPU



CPU



GPU

Caractéristiques d'un GPU

- Beaucoup de cœurs : 448 dans le Nvidia Tesla C2050/C2070 avec une fréquence de 1.15 GHz.
- Support IEEE double précision en virgule flottante
- Mémoire : jusqu'à 6 GB

Exploitation d'un GPU

- Programmation de type SIMD, gestion de milliers de thread
- Transfert CPU-GPU très couteux
- API de programmation (CUDA, OpenCL)

$\rightarrow$ Complexe à exploiter, toutes les applications ne s'y prêtent pas

Evolution ?

- Les machines en tête du top 500 sont des machines mixtes CPU-GPU
- Apparition d'architecture mixte comme le Fusion d'AMD ou le Sandy Bridge d'Intel : packaging d'un processeur et d'un cœur graphique
- Amélioration de l'efficacité des transferts CPU-GPU ?

Conclusions

Fréquence d'horloge, nombre de cœurs, nombre d'unités fonctionnelles, fréquence et capacité mémoire, caches, stockage, interconnexions internes et réseaux ...
Tous les éléments ont leur importance ; l'architecture doit être équilibrée pour que l'ensemble soit performant

Au niveau technologique

- Beaucoup de cœurs
- Des cœurs hybrides CPU/GPU
- Une différence entre la vitesse de calcul et les transferts mémoire et I/O

Conclusions

Au niveau applicatif

- Nécessité d'une bonne connaissance des architectures, notamment au niveau de la mémoire.
- Plus 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 un grand nombre de processeurs.
- Nécessité d'exploiter le parallélisme aux différents niveaux du hardware.
- Programmation sur des architectures hybrides avec différents types de processeurs.

Références

- http://calcul.math.cnrs.fr/
- http://www.realworldtech.com
- https://computing.llnl.gov/tutorials/parallel_comp/
- http://www.idris.fr
- http://www.top500.org
- http://www.prace-project.eu/hpc-training
- http://www.inria.fr/actualites/colloques/cea-edf-inria/index.fr.html
- http://www.nvidia.fr
- http://www.intel.com