Parallélisme: Culture et nouvelles technologies

Jean-Baptiste Keck

Laboratoire Jean-Kuntzmann - Grenoble

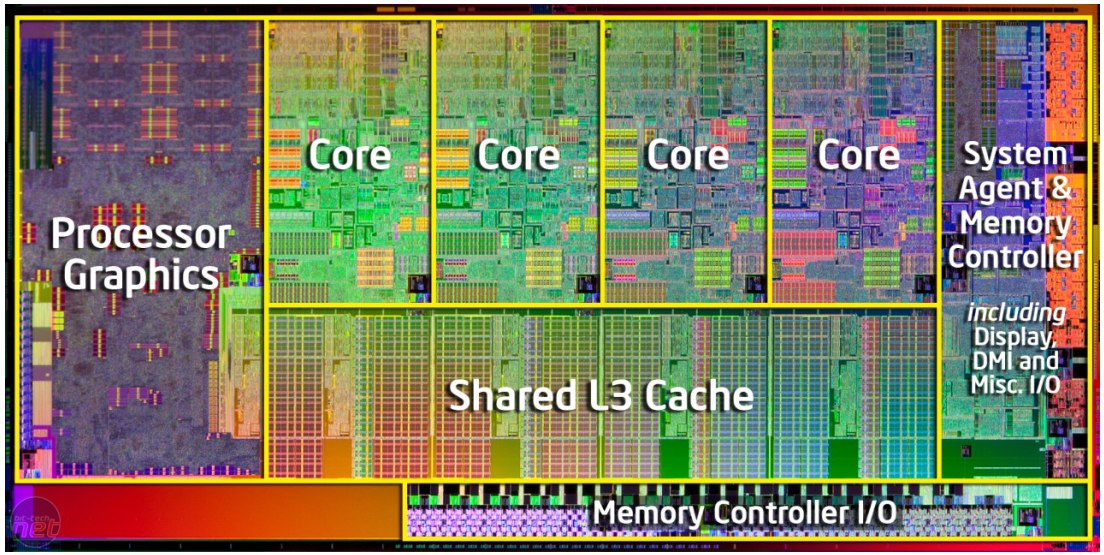
13 octobre 2016

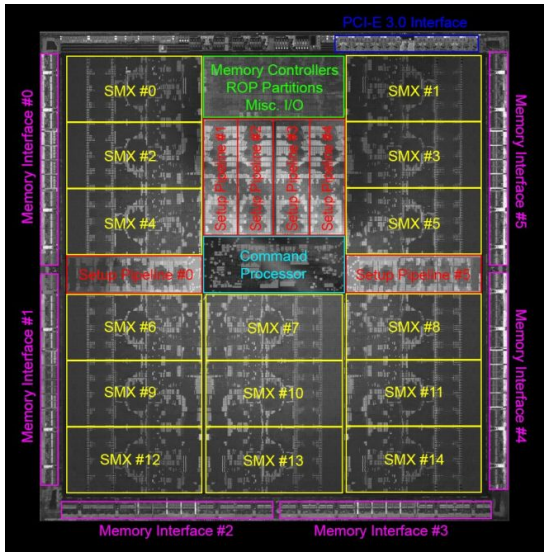
Table des matières

Connaître les architectures : Pourquoi ?

Des architectures de calcul multiples avec des characteristiques différentes

- Comprendre le comportement d'un programme et comment l'architecture et le modèle de programmation affectent les performances de mon code.
- Adapter les algorithmes aux architecture pour extraire le maximum de performance.



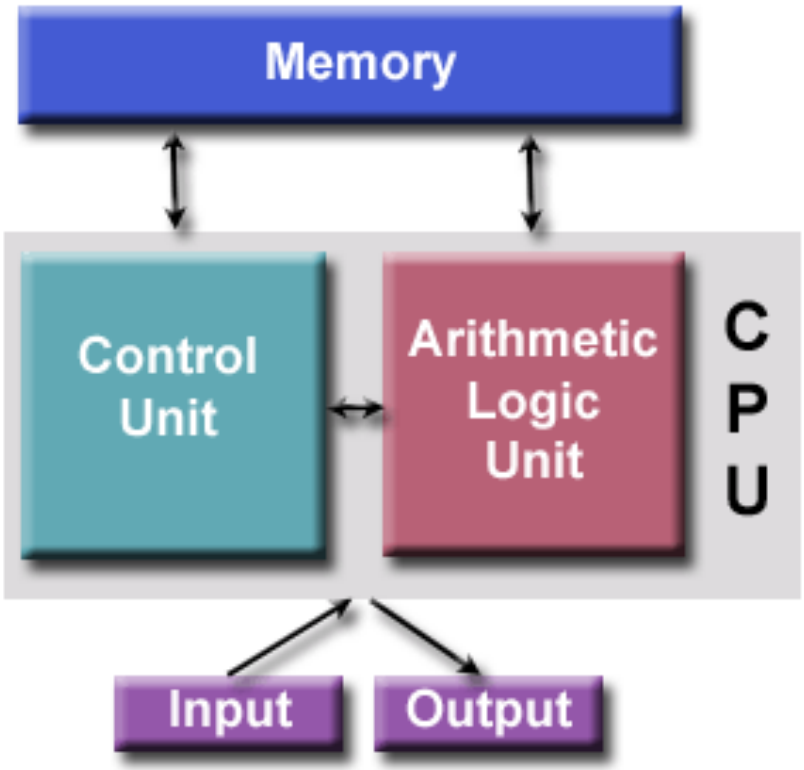


Les élements d'un supercalculateur

- Noeuds de calcul
  - Processeurs (sockets) : fournissent la puissance de calcul
  - Mémoires : permettent de stocker l'executable et ses variables
  - Moyens de communication internes : Bus, canaux mémoires, liens
- Réseaux d'interconnexions : Relie les noeuds entre eux. Souvent de plusieur types et topologies.
- Stockage : Baies, systèmes de fichiers
- Couche logicielle : middleware pour l'accès aux ressources distribuées (MPI, threads, ...)
- Frontale : porte d'entrée du cluster, permettant l'accès au gestionnaire de batch.
- Gestionnaire de batch : Outil permettant de gérer la réservation, l'allocation, l'ordonancement et le placement des ressources d'une machine.

Le modèle de Von Neumann (1945)

Première description d'un ordinateur dont le programme est stocké dans sa mémoire.

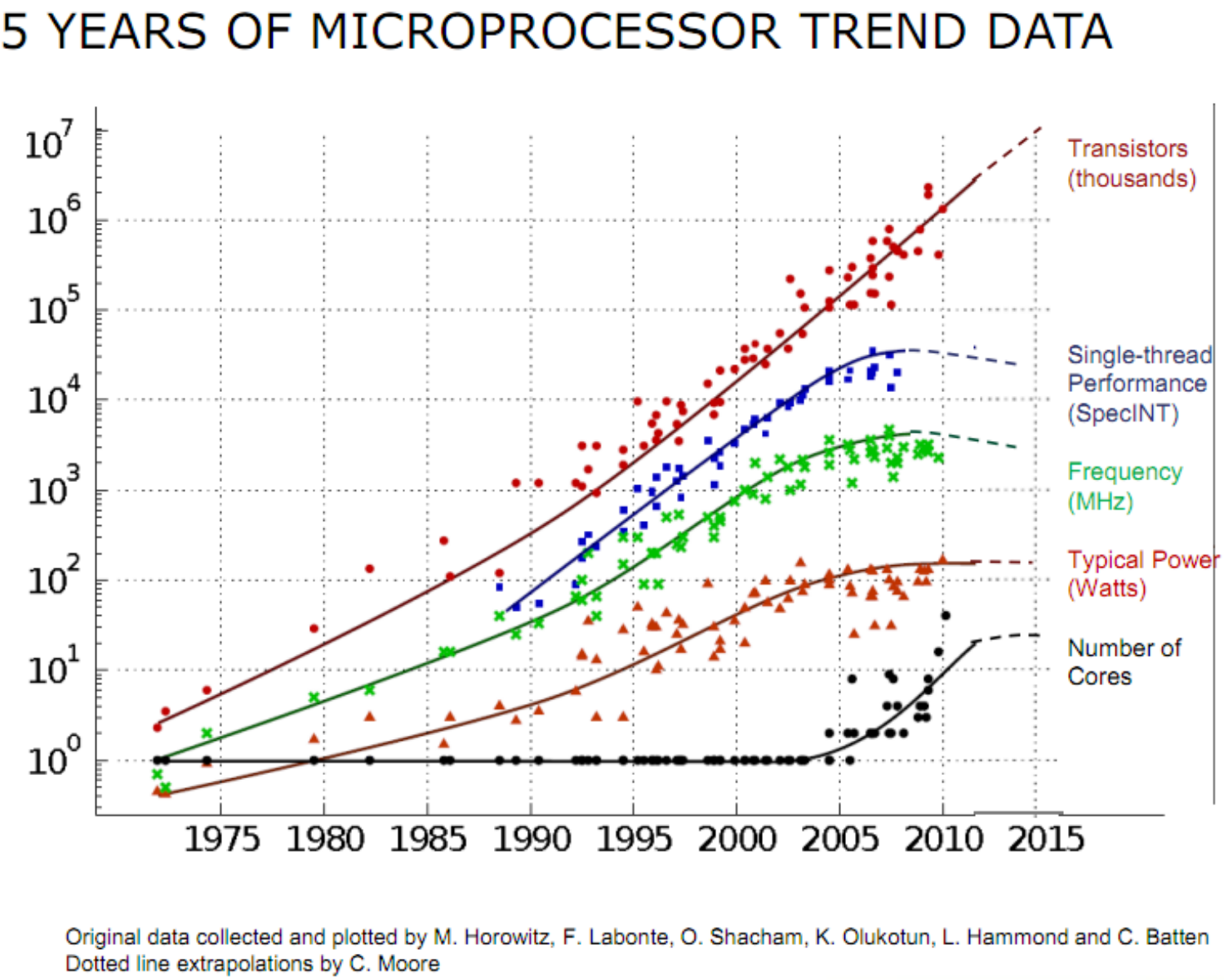


1. Unité arithmétique et logique (ALU) : effectue les opérations de base.
2. Unité de contrôle : Chargée du séquençage des opérations.
3. Mémoire : Contient à la fois les données et le programme (instructions).
4. I/O : Des dispositifs d'entrée-sorties.

Les caractéristiques d'un processeur

- Son jeu d'intruction :
  - CISC (Complex Instruction Set Computer) : beaucoup d'instructions complexes qui prennent plusieurs cycles d'horloge (x86,x64,...)
  - RISC (Reduced Instruction-Set Computer) : moins d'instructions n'utilisant que quelques cycles (PowerPC, MIPS, ARM)
- La taille de ses registres (taille des "mots") : Souvent 32 ou 64bits.
- Sa fréquence d'horloge :
  - La fréquence d'horloge détermine la durée d'un cycle.
  - Chaque instruction utilise un certain nombre de cycles.
  - La consommation électrique croit en F^3.
- Son nombre de transistors : Dépend de sa finesse de gravure et de la surface occupée.
- Son nombre de coeurs : Présence ou pas d'un certain de coeurs de calcul indépendants.

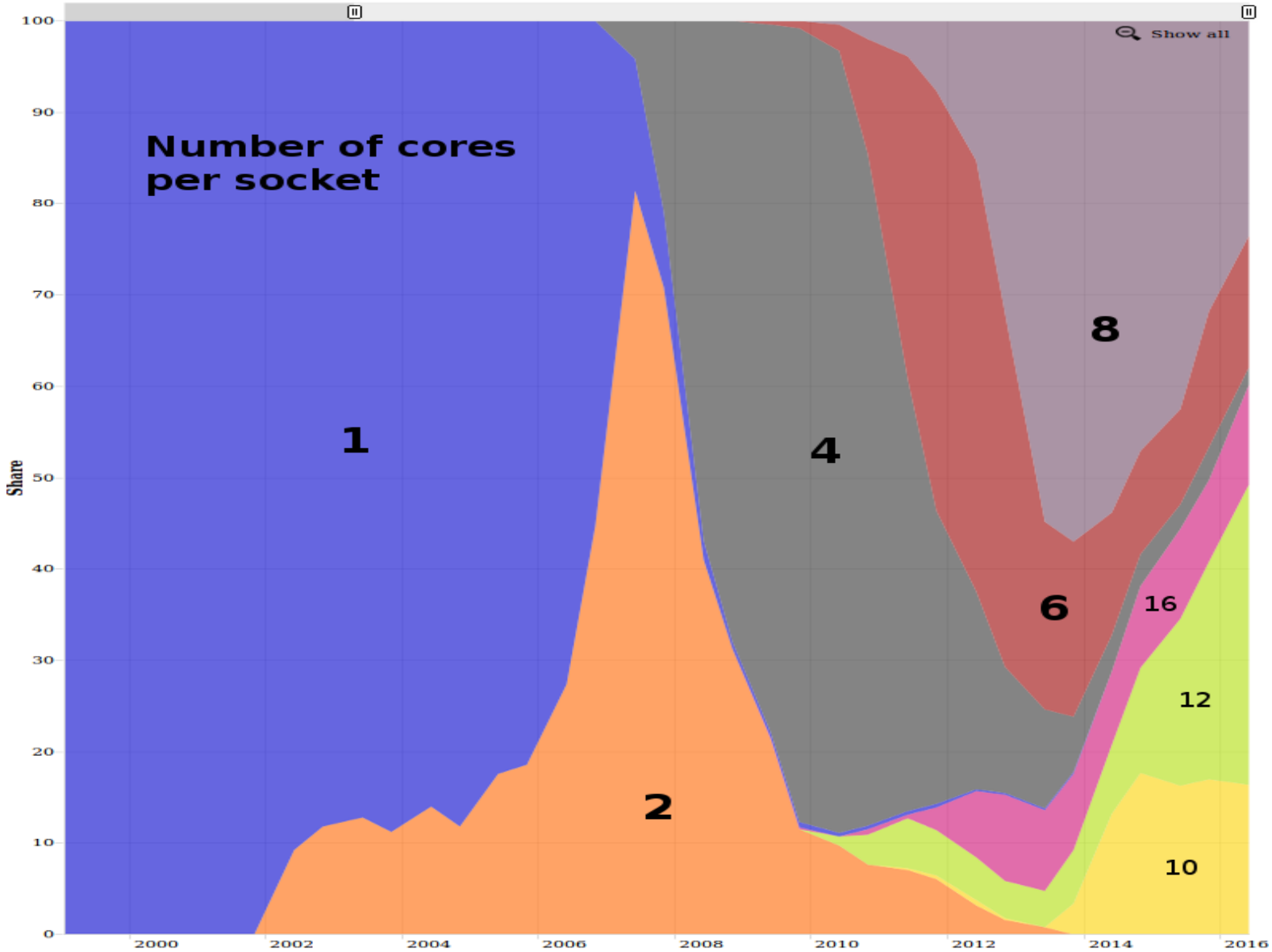
La tendance depuis les années 70



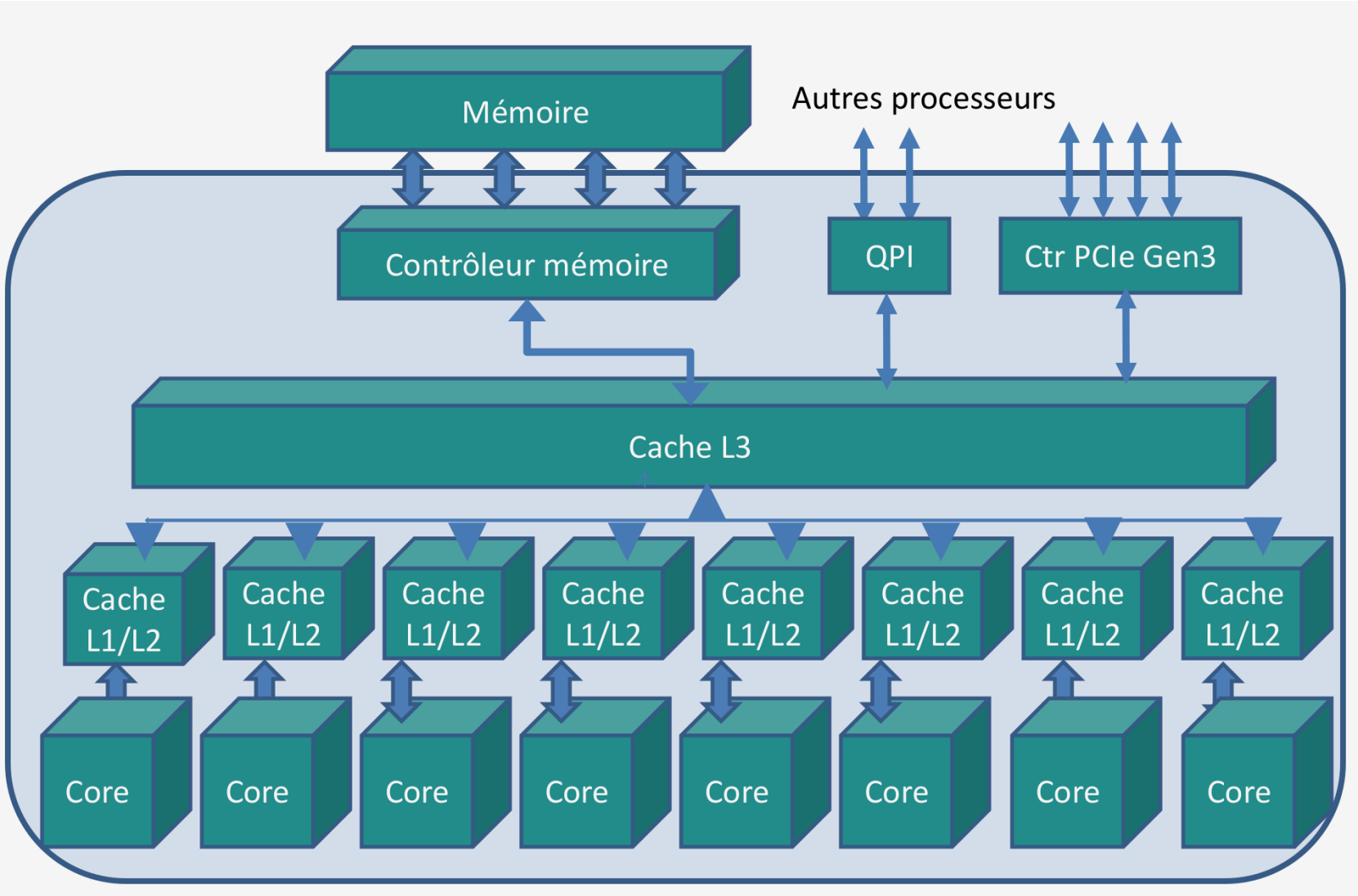
Mesure de performances

- Métrique : On utilise l'opération en virgule flottante par seconde (FLOPS ou flop/s)
- TOP500 : Liste des 500 ordinateurs les plus rapides, actualisée tous les 6 mois et basé sur un benchmark Linpack (FLOPS).
- GREEN500 : Liste des 500 ordinateurs les plus performants au niveau énergétique (FLOPS/W).

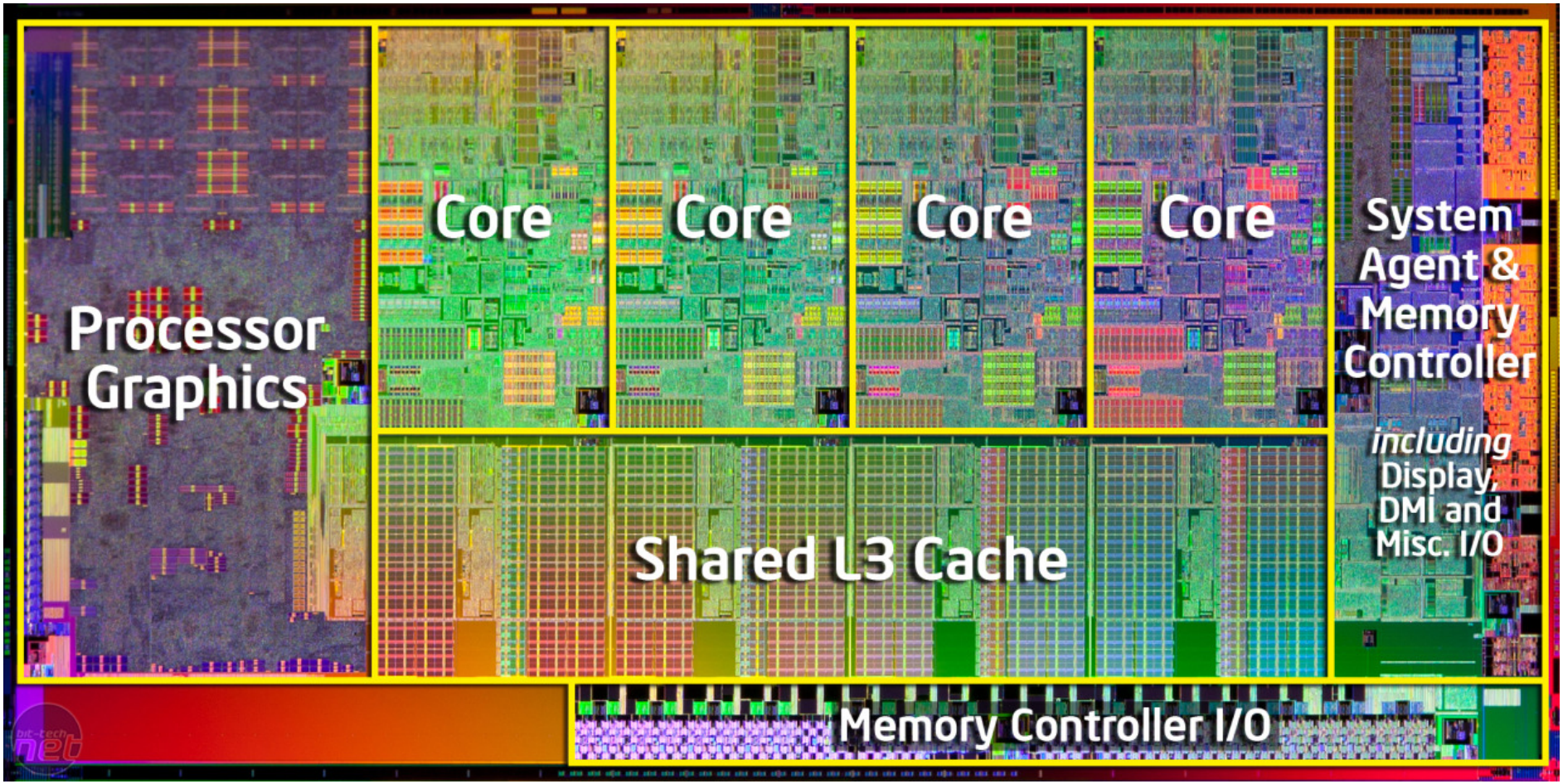
La tendance dans les supercalculateurs (TOP500)



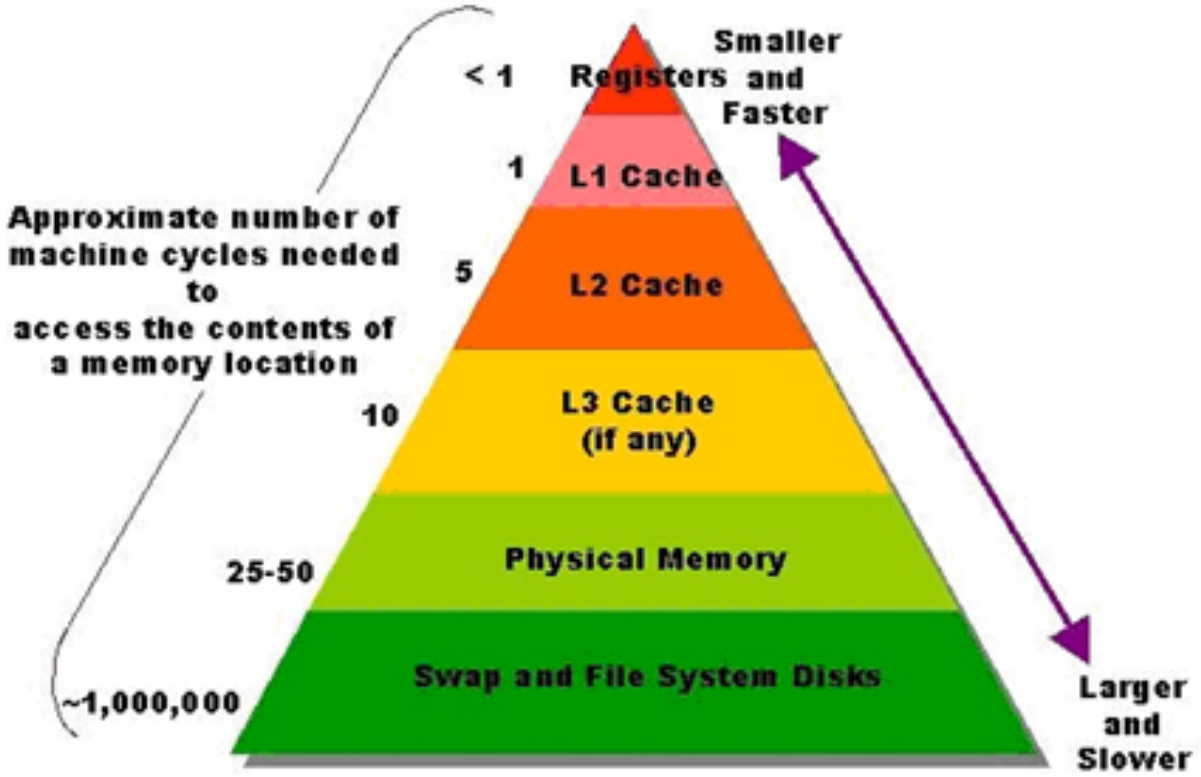
Un processeur en 2016



Un processeur en 2016



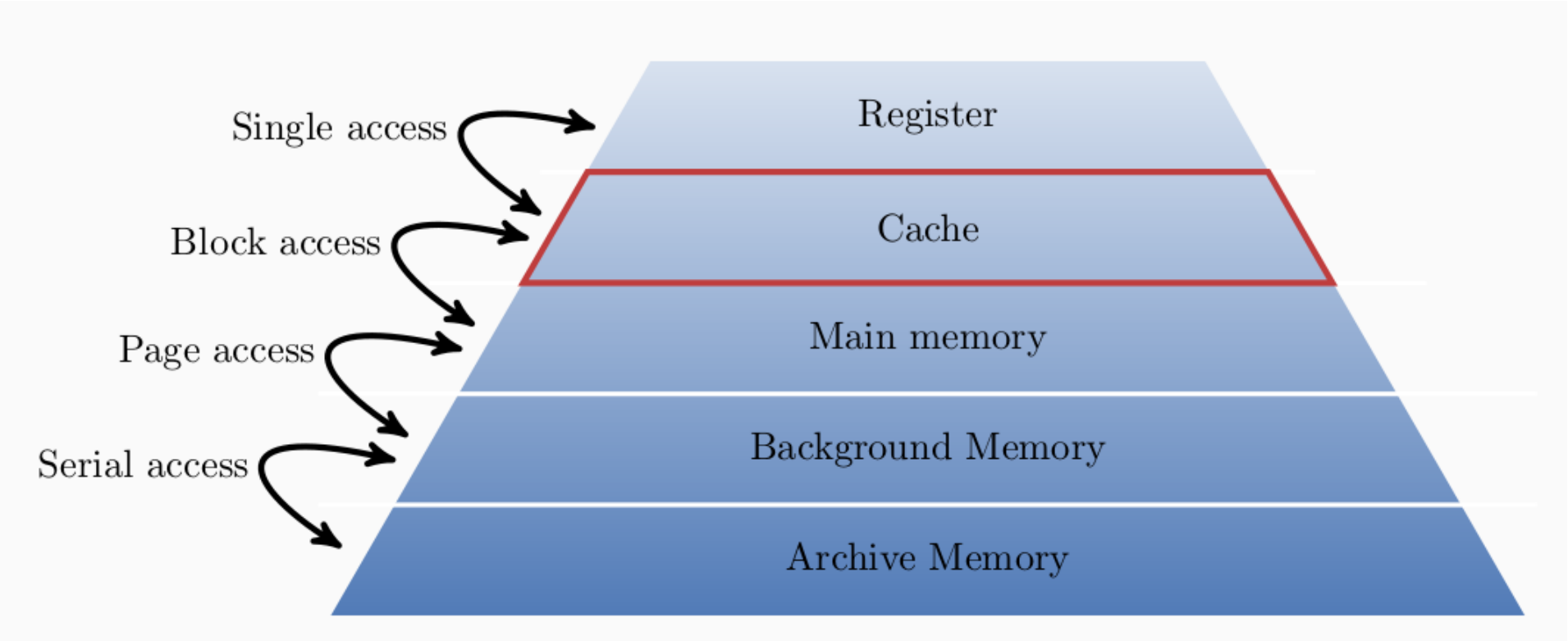
Hiérarchisation de la mémoire



Les métriques

- Latence : temps d'accès à une donnée pout des accès aléatoires.
- Débit : la bande passante (Go/s) pour des accès séquentiels.

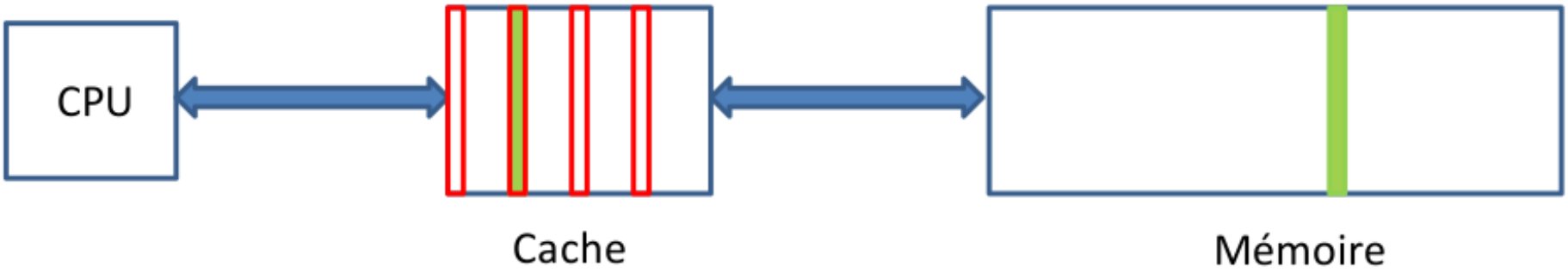
Des accès avec différents niveaux de granularité



- Cohérence : Une donnée peut être présente à différents niveaux de la mémoire, ce qui pose des problèmes de cohérence de de propagation.
- Cache : Si le processeur tente d'accéder à une information en cache, le processeur y accède quasiment sans aucun délais (cache-hit).

Pourquoi les caches fonctionnent-ils ?

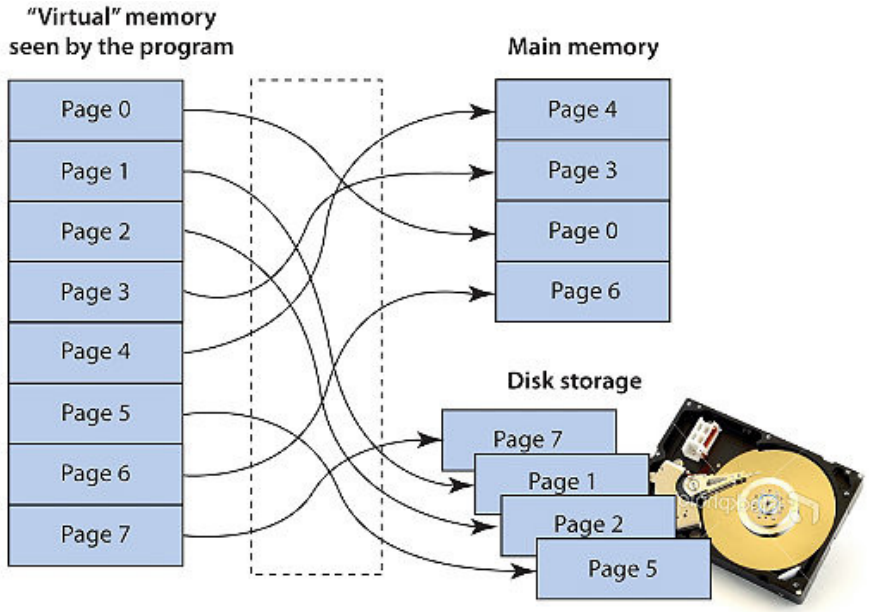
- Localité temporelle : Une donnée référencée à un temps t a de grandes chance d'être référencée à nouveau dans un futur proche.
- Localité spatialle : Si une donnée est référencée, les zones voisines ont de grandes chance d'être référencées dans un futur proche.



La mémoire virtuelle

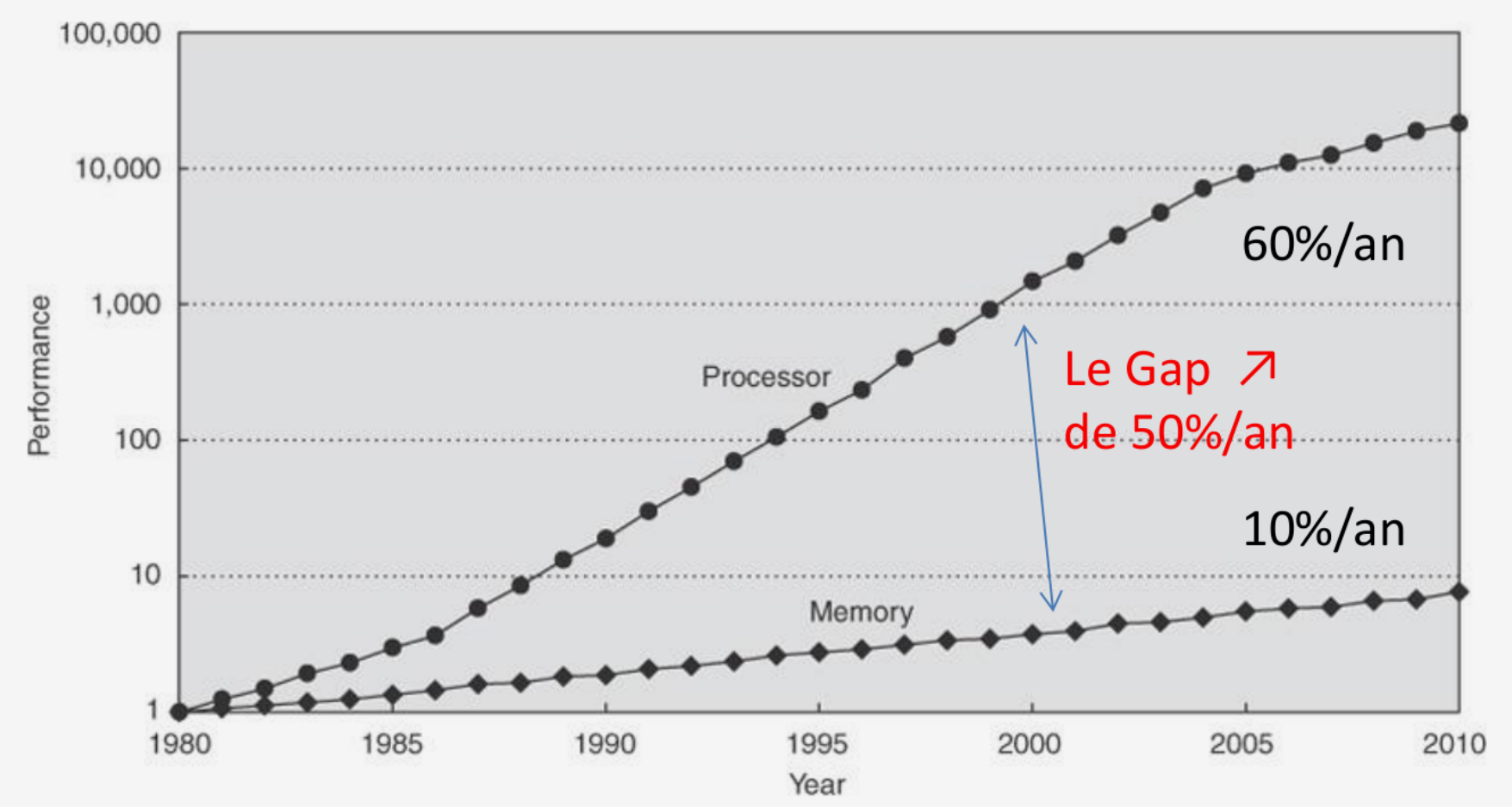
La mémoire virtuelle est le dernier niveau de la hierarchie mémoire

- Un programme utilise des adresses virtuelles.
- Il faut traduire les adresses virtuelles en adresses physiques.
- L'espace d'adressage virtuel de chaque programme est divisé en pages.
- Une unité materielle est chargée de faire la correspondance, c'est la MMU (Memory Management Unit).



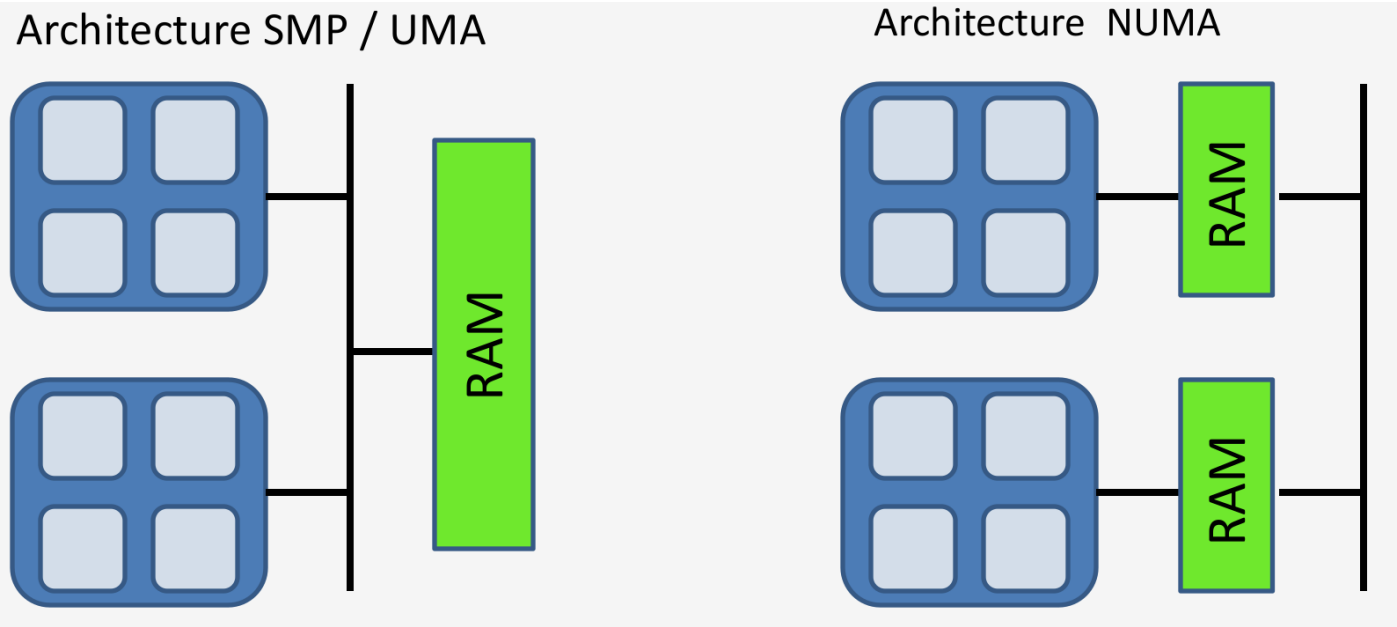
Performance processeur vs performance mémoire

La performance des ordinateurs est surtout limitée par les accès mémoire (temps d'accès mémoire >> temps d'un cycle processeur).

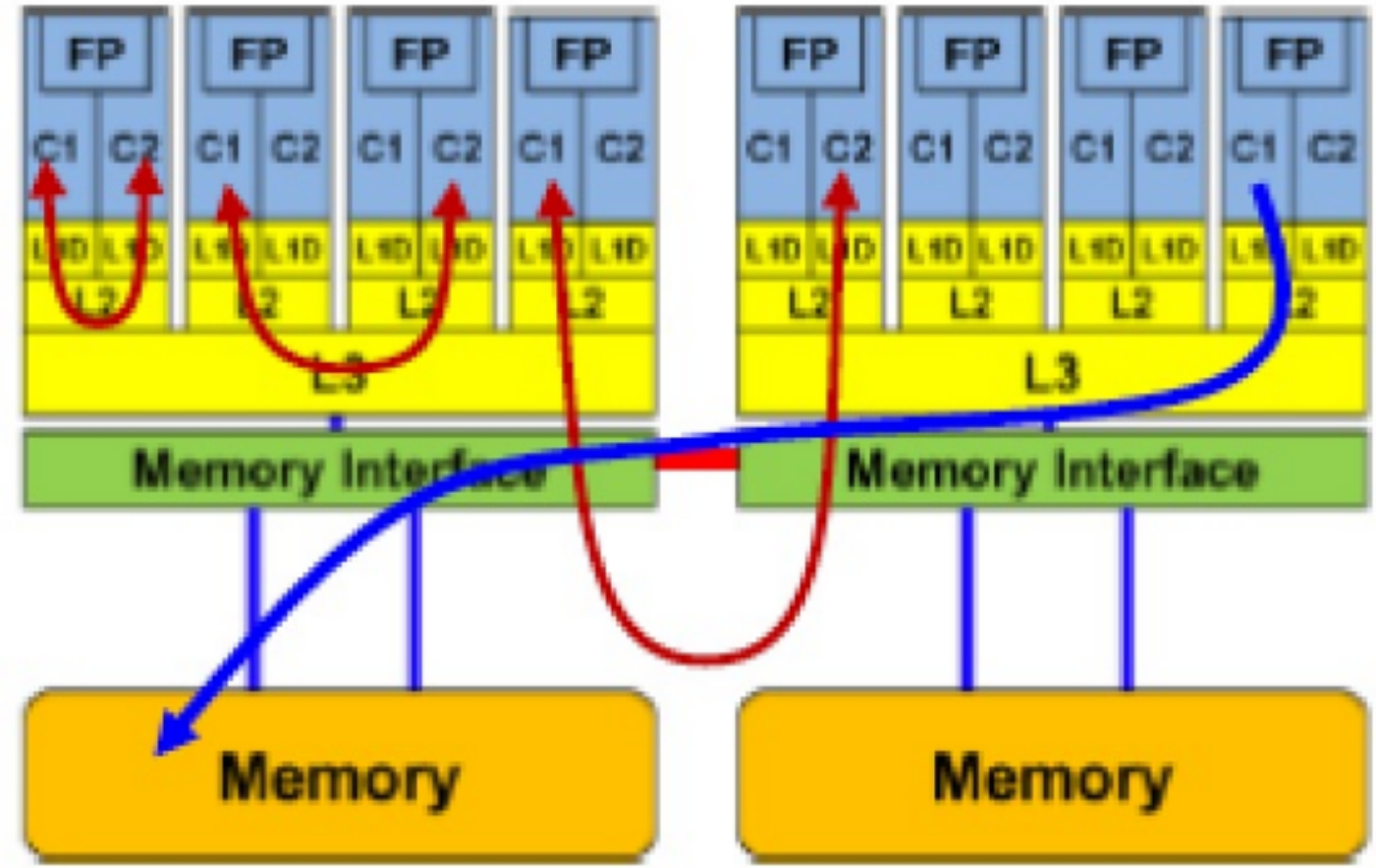


Modèles d'architectures partagées

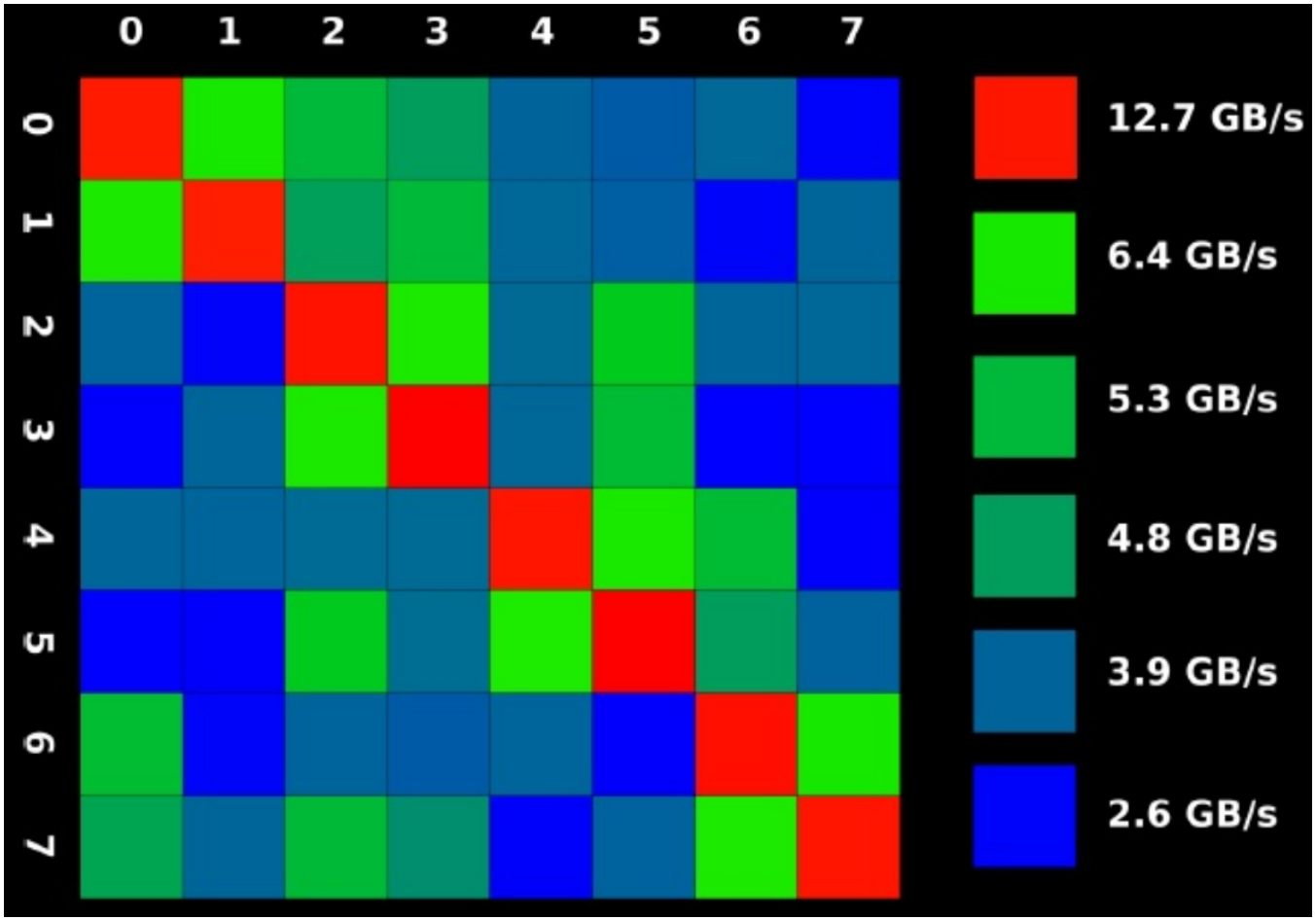
- Architecture UMA (Uniform Memory Access) : Temps d'accès à un emplacement mémoire quelconque identique pour les les processeurs.
- Architecture NUMA (Non Uniform Memory Access) : Les processeurs peuvent accéder à la totalité de la mémoire les mais temps d'accès peuvent différer selon la distance coeur-mémoire.



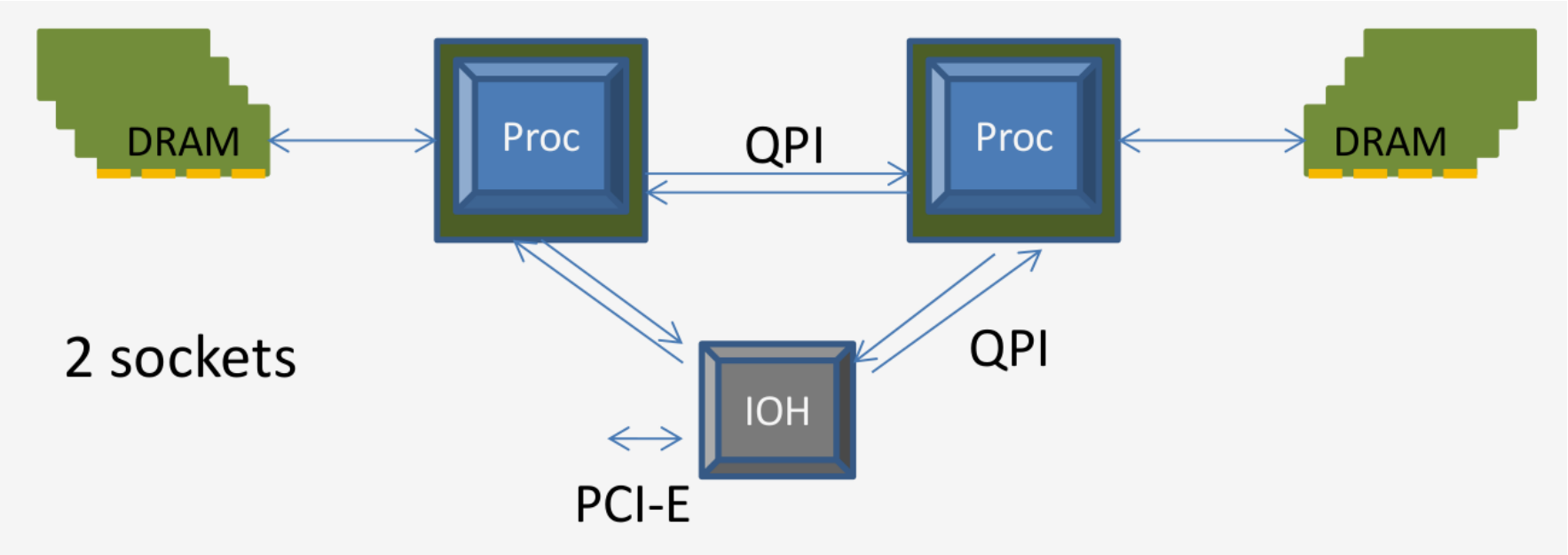
Exemple de communication asymétrique NUMA



Exemple de communication asymétrique NUMA

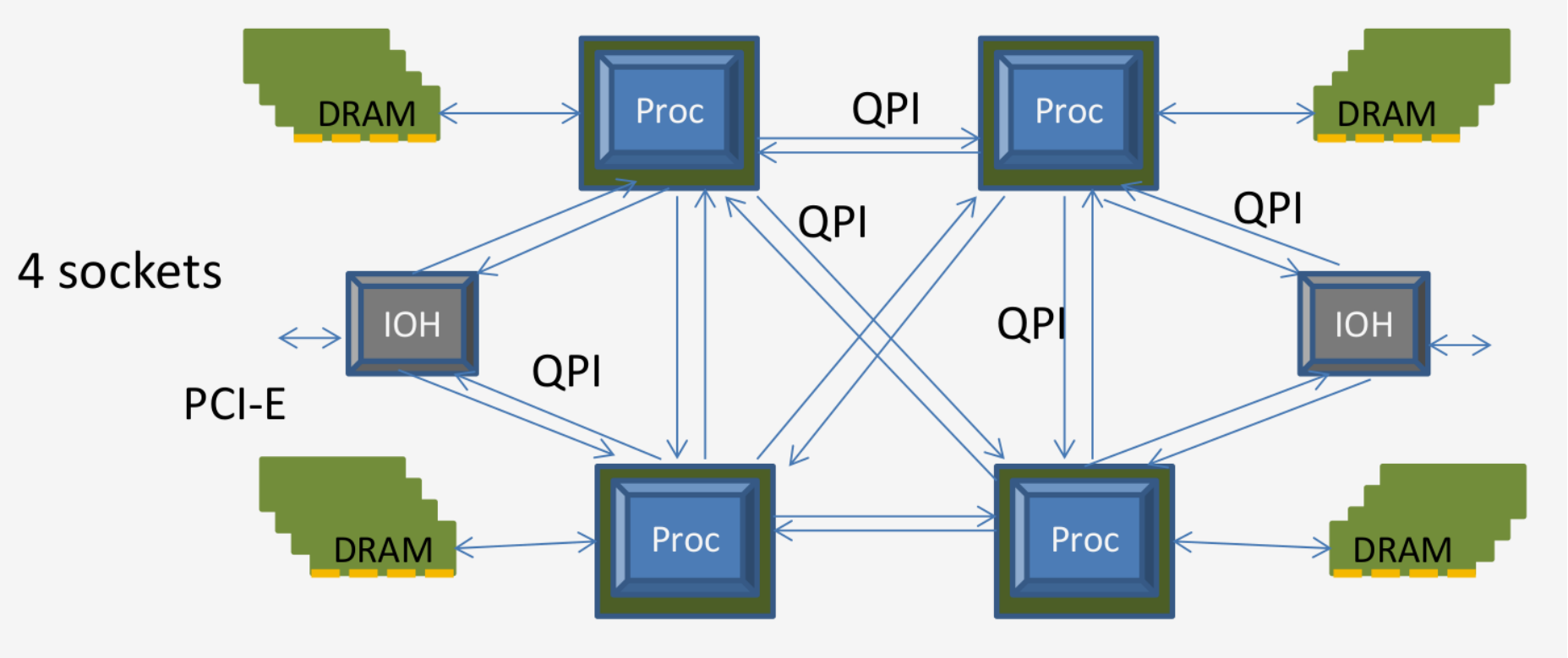


Machines à plusieurs sockets



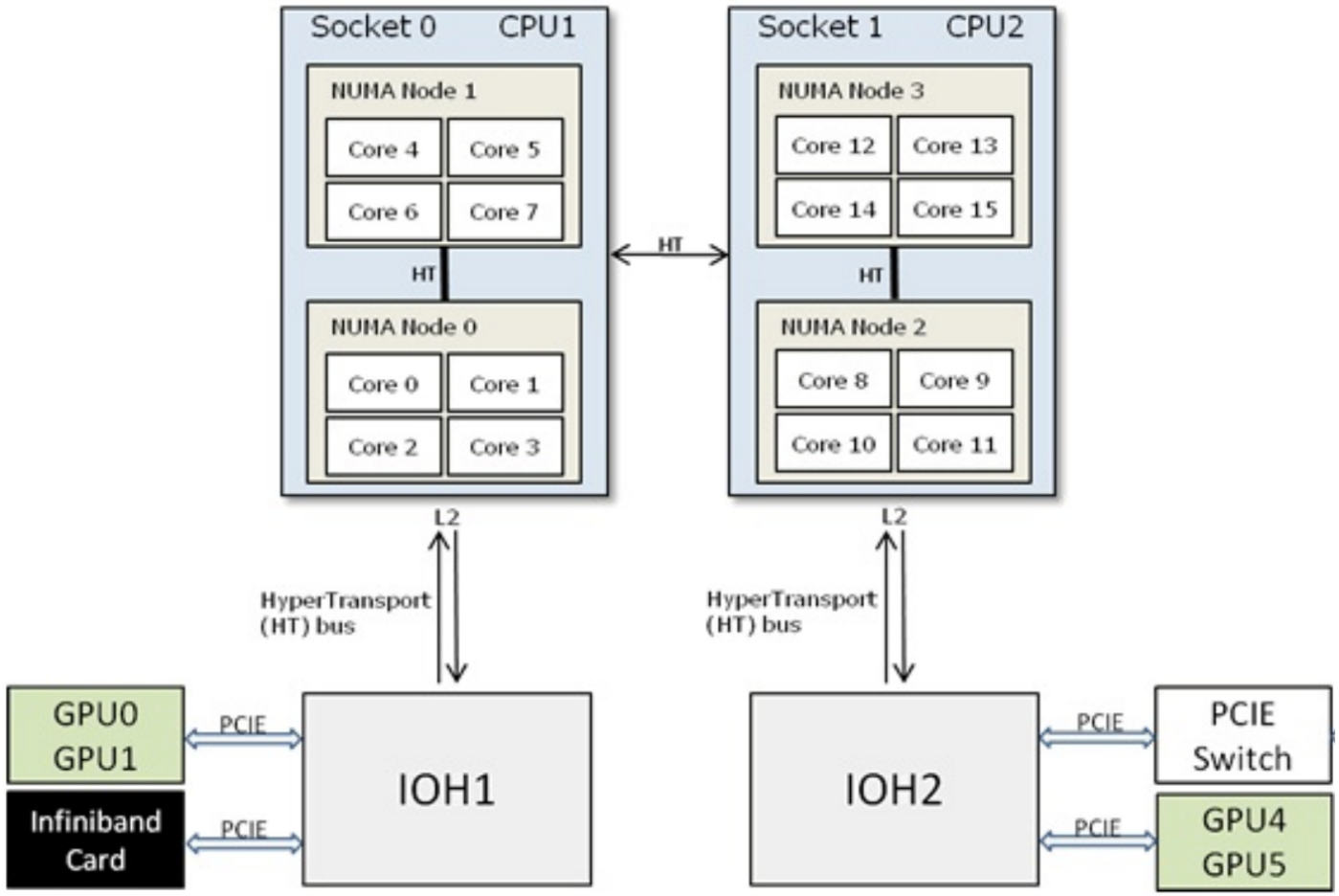
Noeud NUMA à deux sockets

Machines à plusieurs sockets

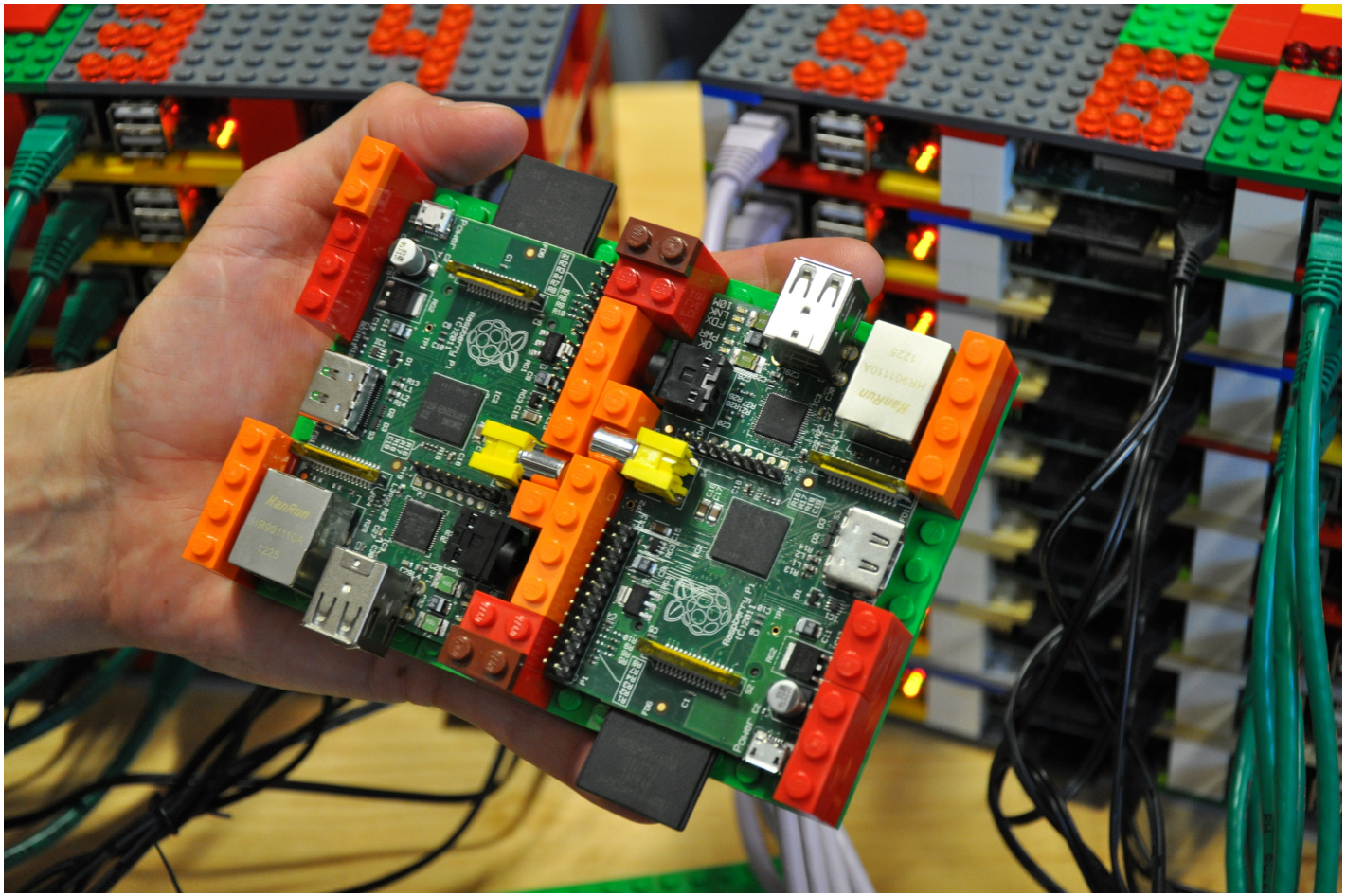


Noeud NUMA à quatre sockets

Vue globale d'un noeud de calcul



Les fermes de calcul (cluster)

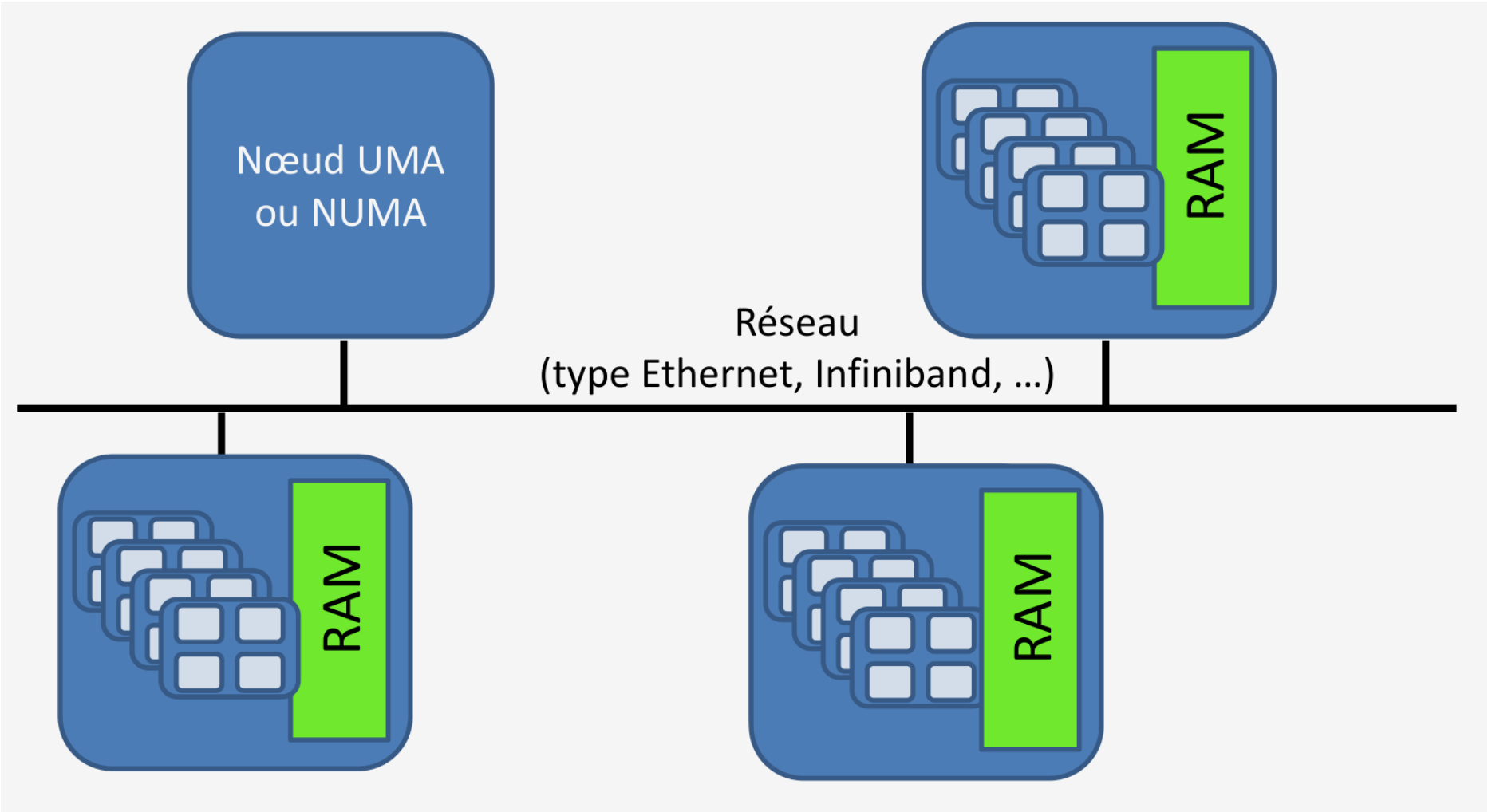


Cluster de raspberry pi

Les fermes de calcul (cluster)

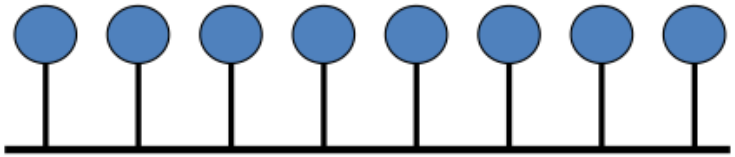
Tianhe-2 (TOP#2, 3,120,000 coeurs de calcul)

Architecture à mémoire distribuée

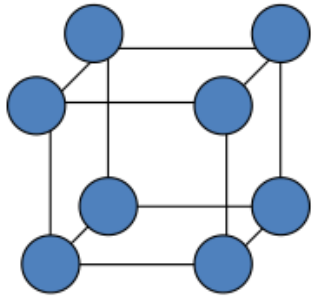


Topologie des réseaux

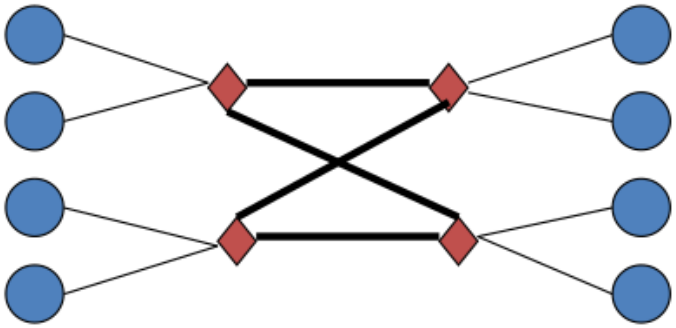
Bus



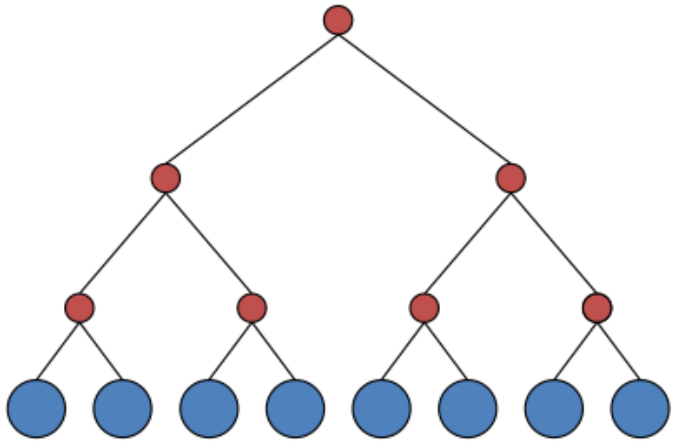
Hypercube



Réseau à plusieurs étages



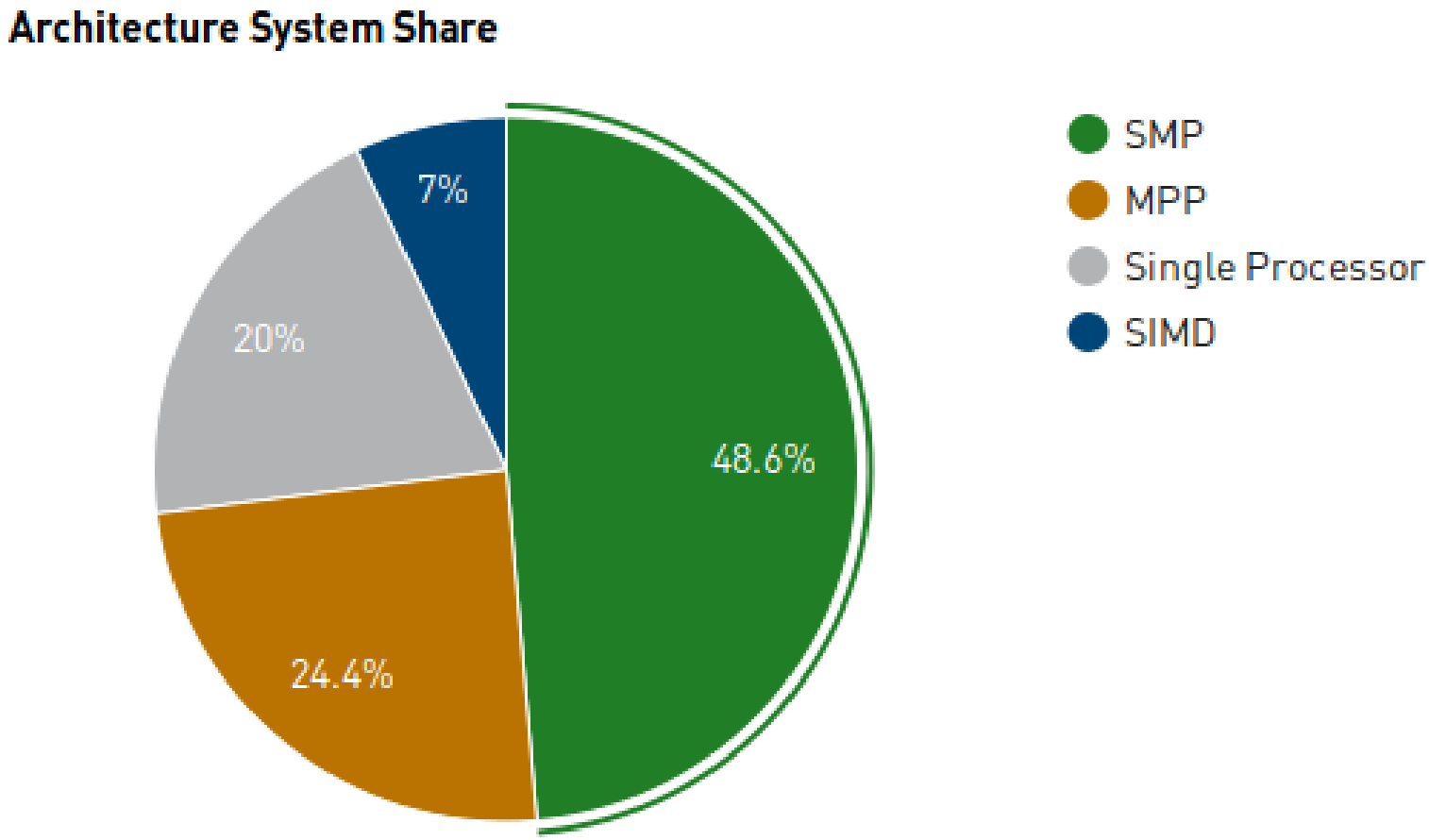
Arbre



Technologies réseau pour le HPC

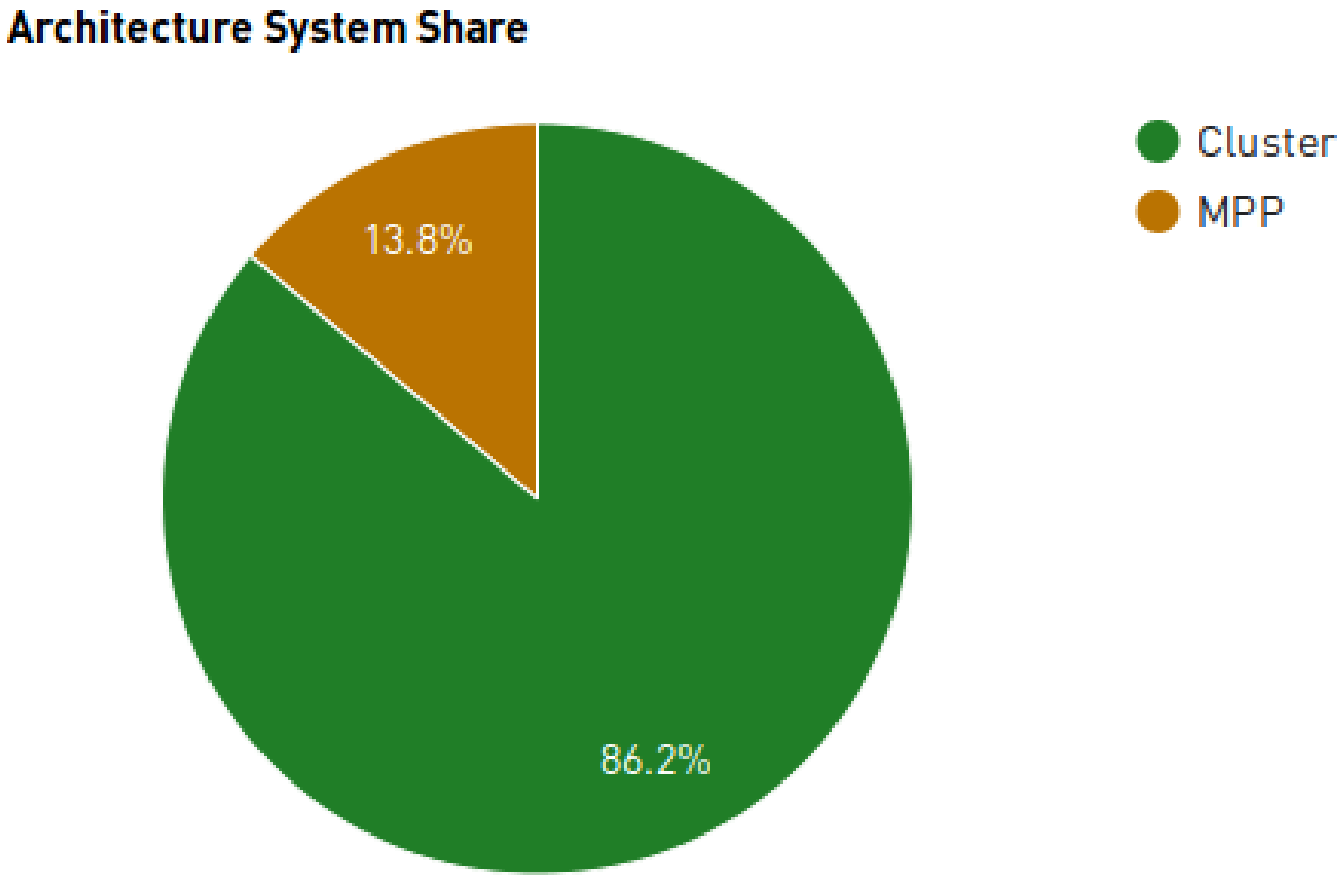
Technologie	Vendeur	Latence MPI usec, short msg	Bande passante
NUMAlink 5	SGI	~1	15 Go/s
QsNet II	Quadrics	~1	900 Mo/s
Infiniband	Mellanox, Qlogic, (Voltaire)	~1	SDR (2.5 Gb/s), DDR (20 Gb/s), QDR (40 Gb/s), FDR (56Gb/s)
Myri-10G	Myricom	~2	1.2 Go/s
Ethernet 10 Gb		~10	10 Gb/s
Ethernet 1 Gb		~50	1 Gb/s

Distribution des architectures partagées dans le TOP500



1993

Distribution des architectures partagées dans le TOP500



2016

Evolution de la performance depuis les années 1995

1 Architectures

2 Concepts du parallélisme
- Principes de base
- Parallélisme d'instructions
- Parallélisme de données
- Parallélisme de processus
- Standard actuel : MPI (1993)
- Modèle des threads
- Parallélisme hybride

3 Les accélérateurs

4 Conclusion

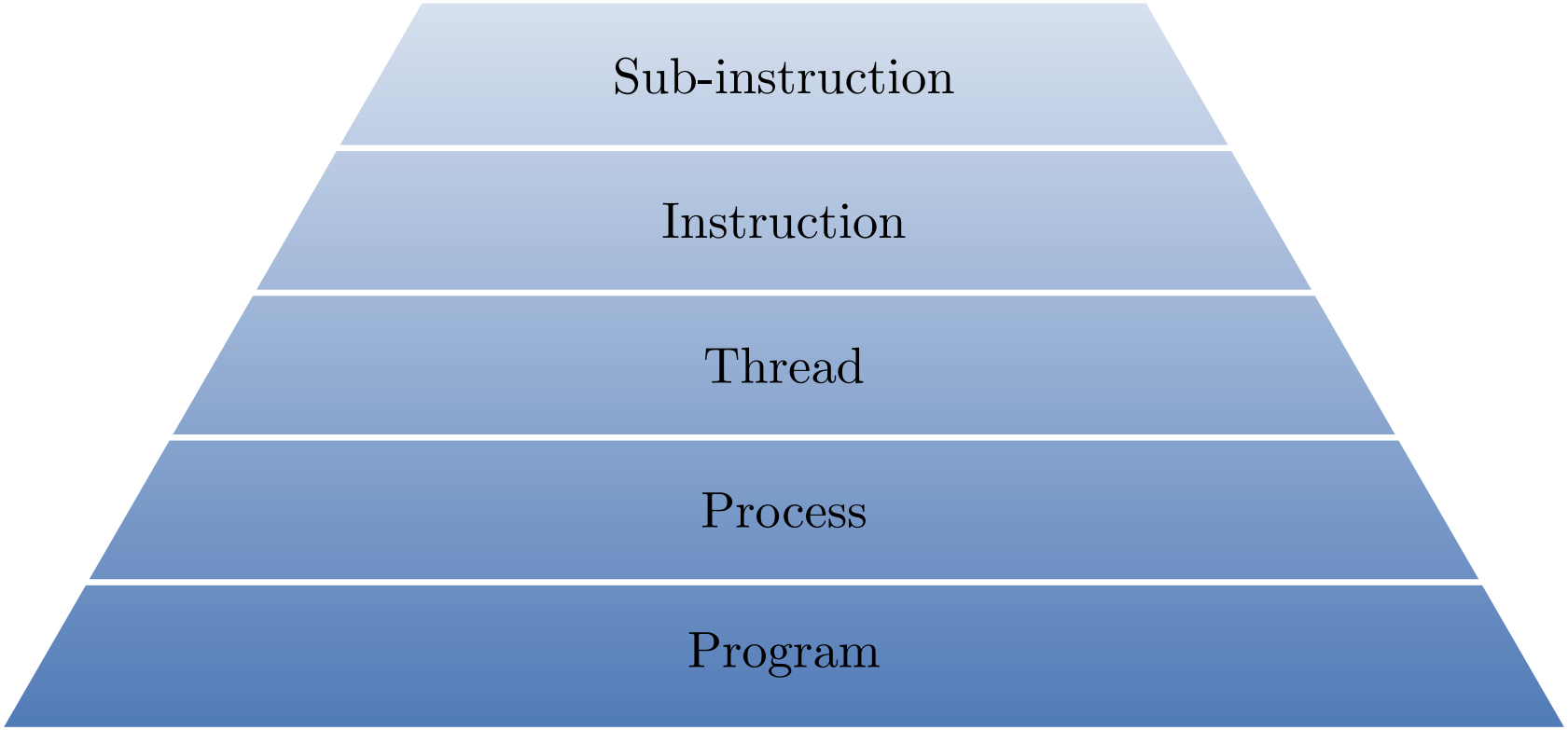
Comment augmenter la performance ?

- Augmenter la fréquence d'horloge (solution couteuse en énergie).
- Permettre l'execution simultanée de plusieurs instructions :
  - Instruction Level Parallelism : Pipelining
  - Thread Level Parallelism : Multithreading, multicore et multinoeuds.
  - Parallélisme de données : Instructions superscalaires
- Améliorer les accès mémoire : Accès aux différents niveaux de cache.

Pourquoi utiliser le paraléllisme ?

- Dépasser les limites économiques et technologiques du séquentiel (frequence, bande passante, miniaturisation, coût).
- Réduire les temps d'éxecution pour un même problème.
- Résoudre des problèmes de plus grande taille.
- Résoudre plusieurs problèmes en même temps.

Qu'est ce qu'un programme ?

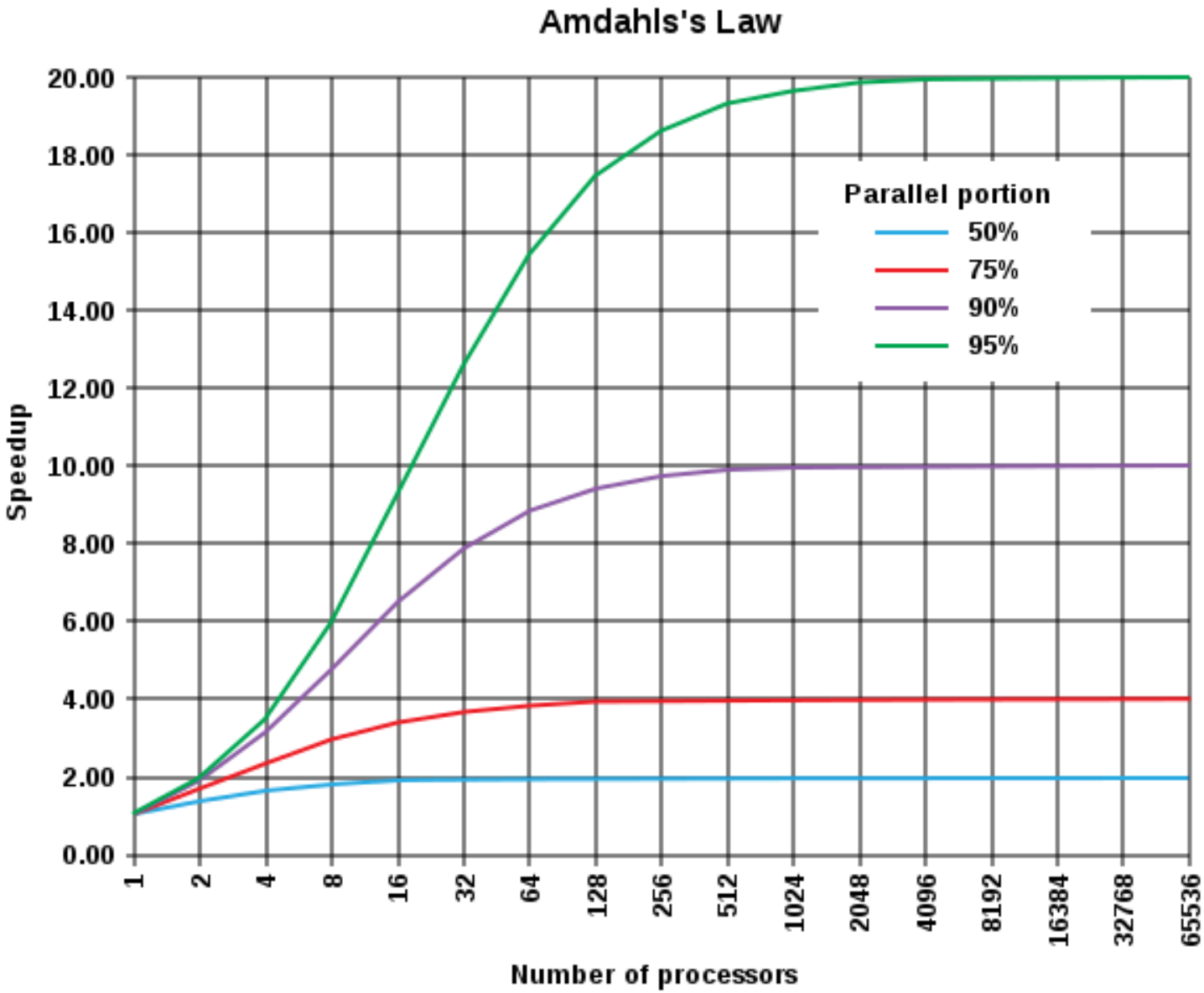


Programme parallèle : Accélération et efficacité

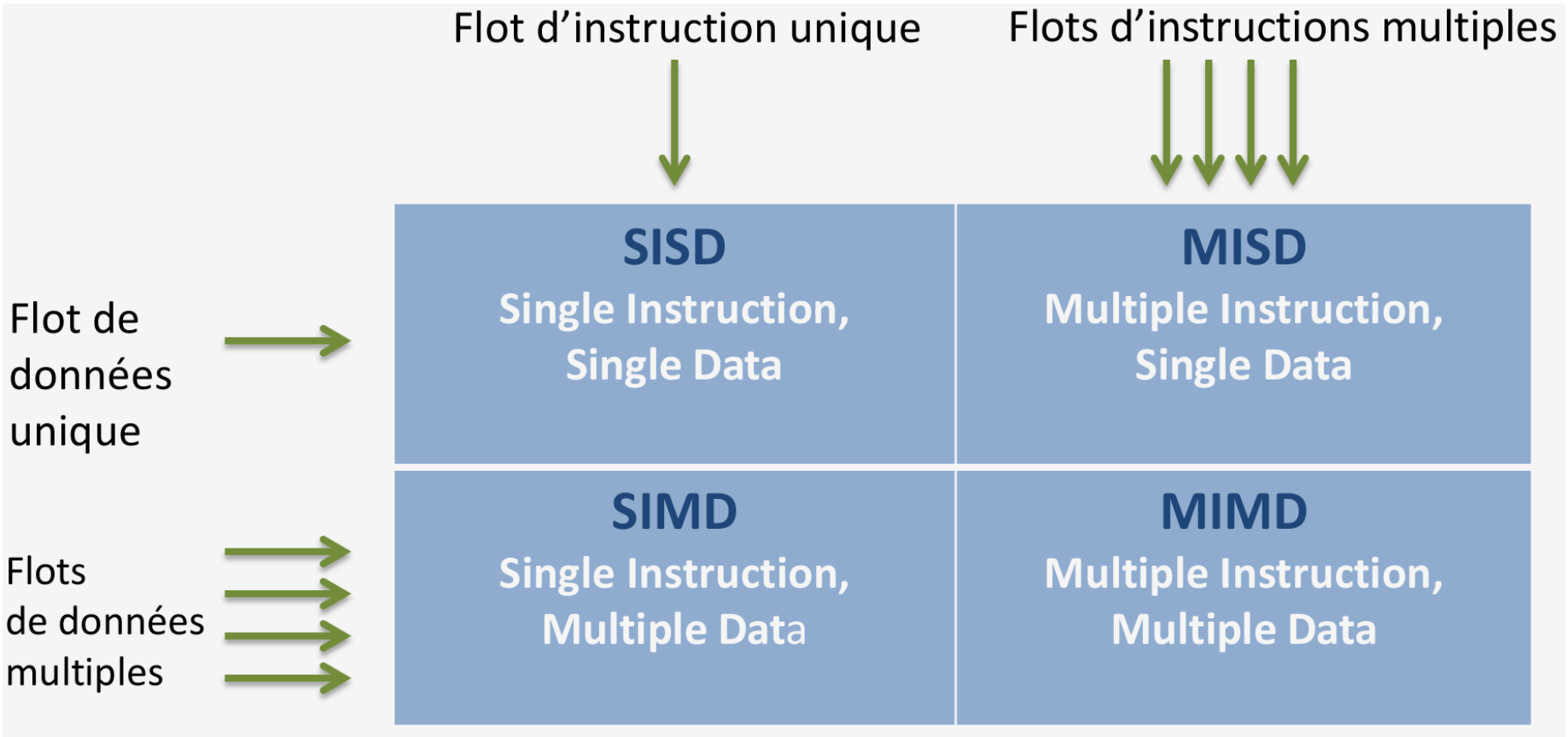
Soit s la faction séquentielle d'un programme, la fraction parallèle est $1 - s$.

- Scalabilité : Propriété d'une application à être éxecutée efficacement sur un grand nombre de ressources (coeufs, sockets, noeuds).
- Acceleration : $A(N) = T_{sequentiel} / T_{Ncores}$.
- Acceleration maximale : Loi d'Amdhal $\Rightarrow A_{max}(N) = \frac{1}{\frac{1-s}{N} + s}$.
- Limites dues aux communications inter-noeuds : $T_{communication} / T_{calcul}$ est un paramètre important.

Loi d'Amdhal


Amdahls's Law
Speedup
20.00
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
Parallel portion
50%
75%
90%
95%
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
Number of processors

Classification des machines parallèles (Flynn)



Execution d'une instruction

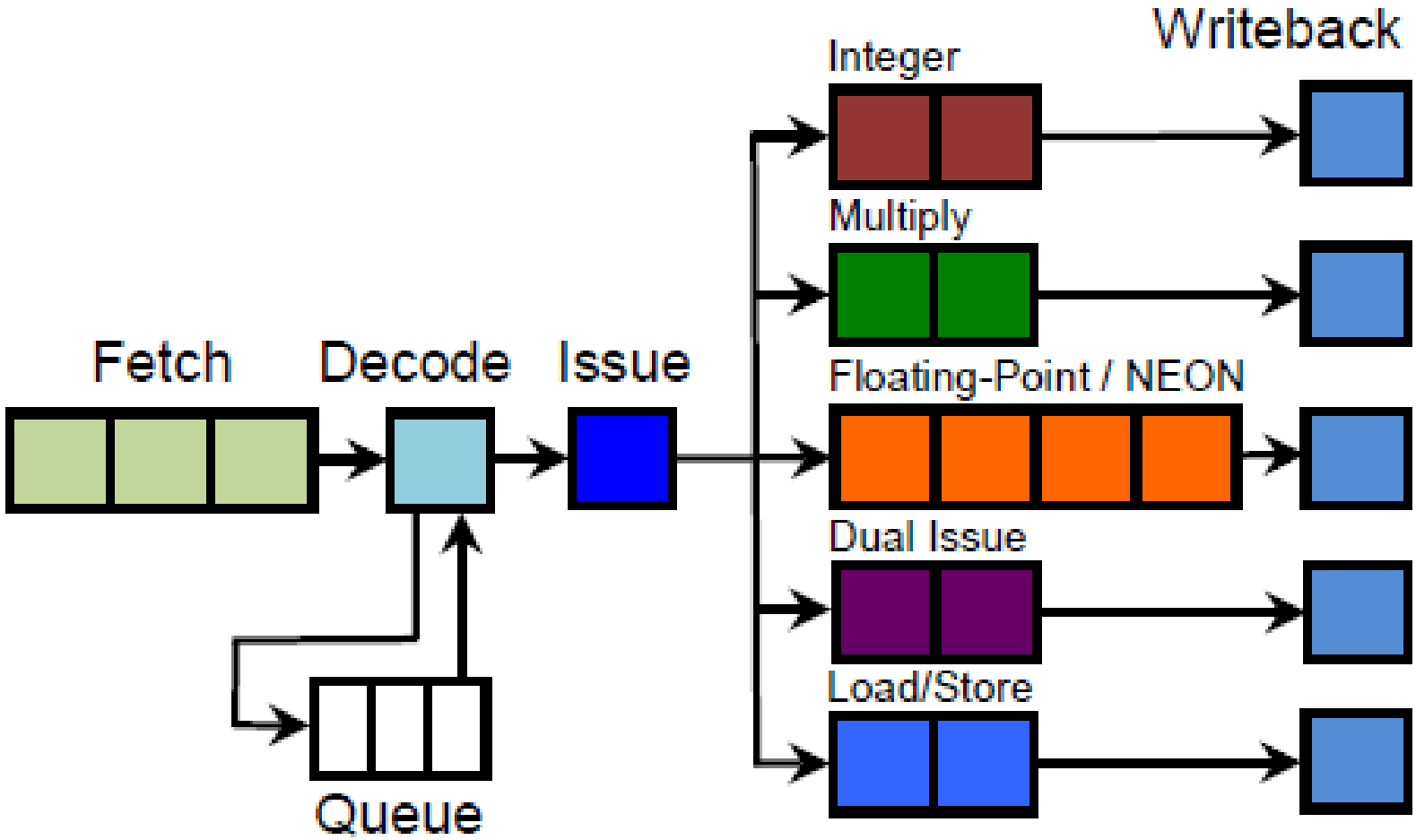
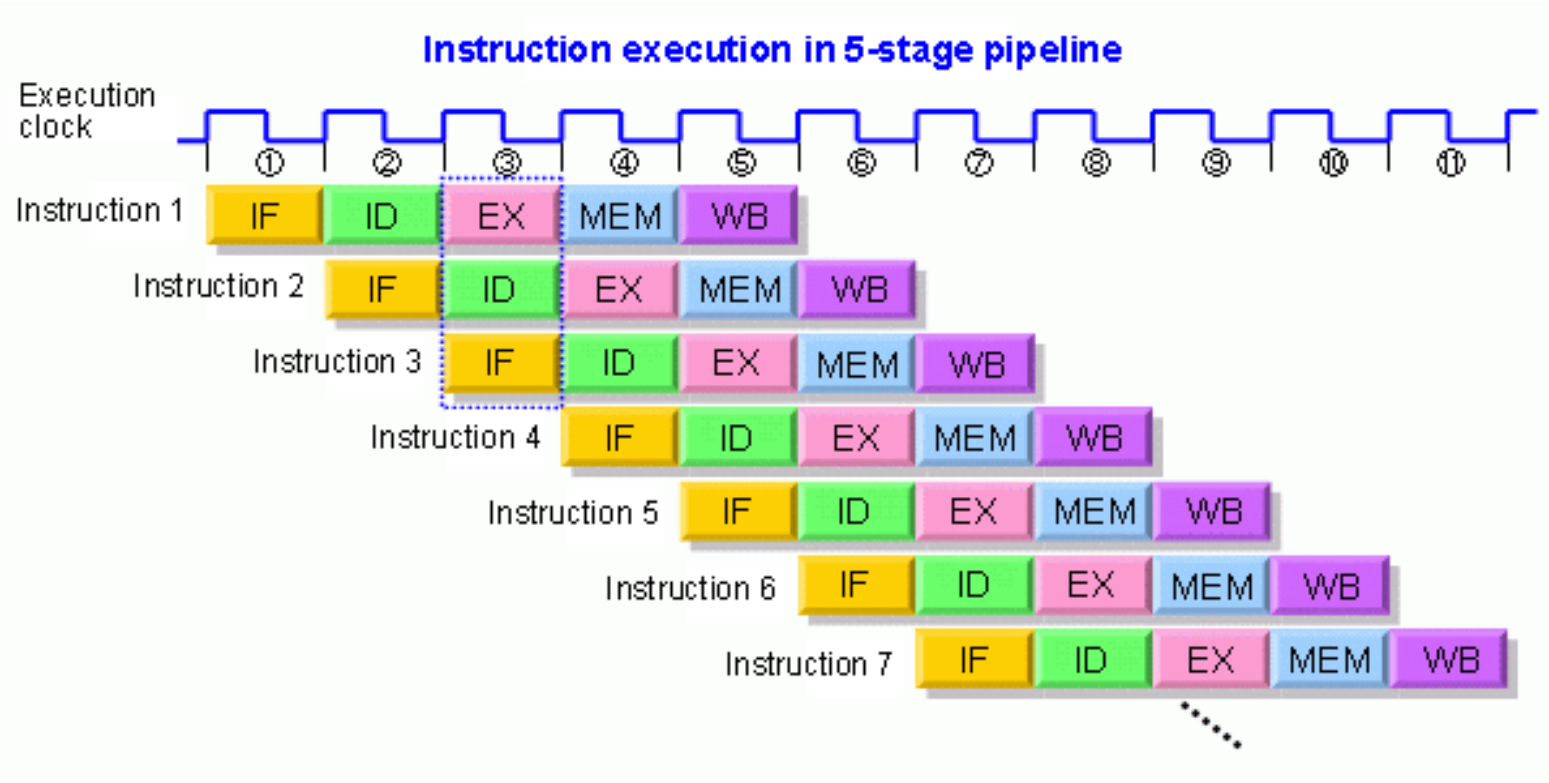


Figure 1 Cortex-A7 Pipeline

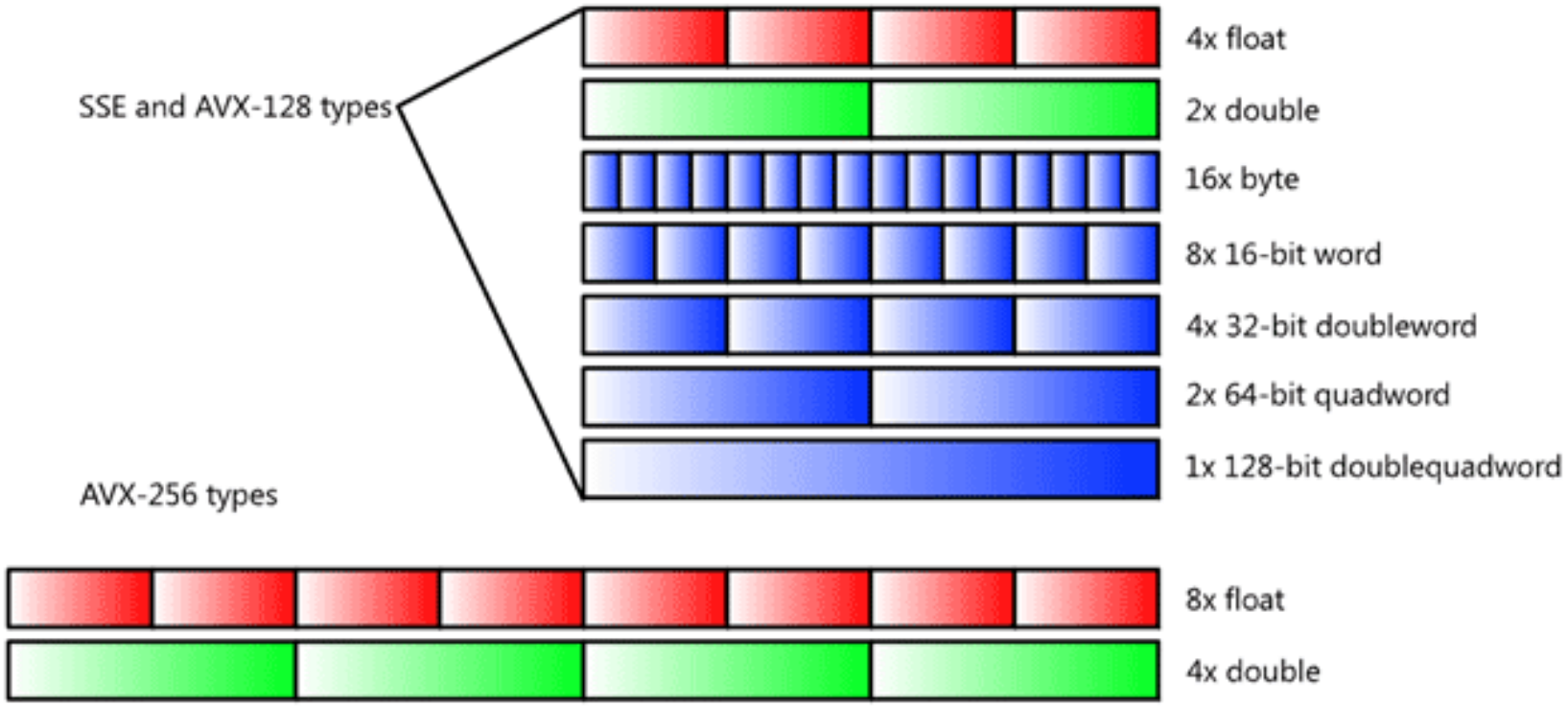
Pipelining : Multiplication des unités materielles



- Execution dynamique : Le hardware modifie l'ordre des opérations pour favoriser l'occupation des ressources.
- Execution spéculative : Le hardware fait une hypothèse sur la suite d'un traitement après un branchement conditionnel.

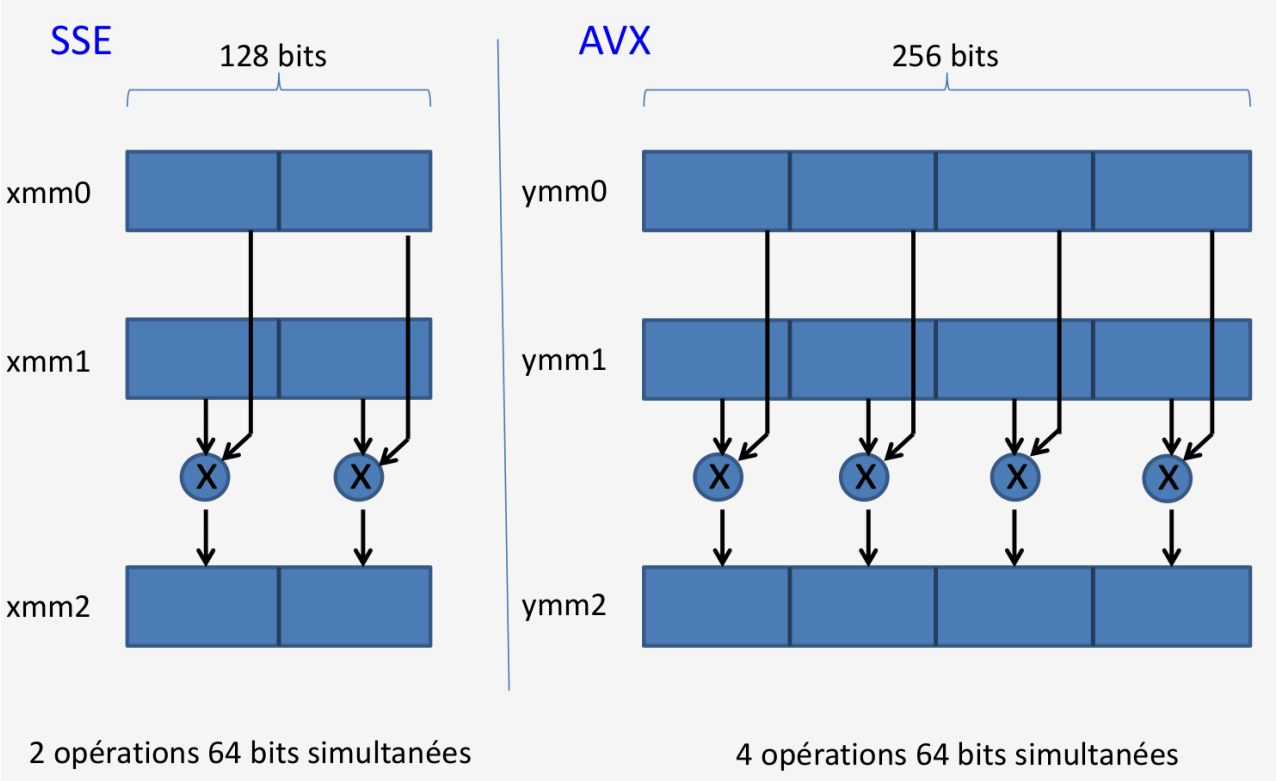
Parallélisme de données

- Instructions superscalaires : Possibilité d'éxecuter une opération sur des registres vectoriels.
- Implémentations : AVX (Advanced Vector Extensions), SSE



Parallélisme de données

- Instructions superscalaires : Possibilité d'éxecuter une opération sur des registres vectoriels.
- Implémentations : AVX (Advanced Vector Extensions), SSE



Processus lourd

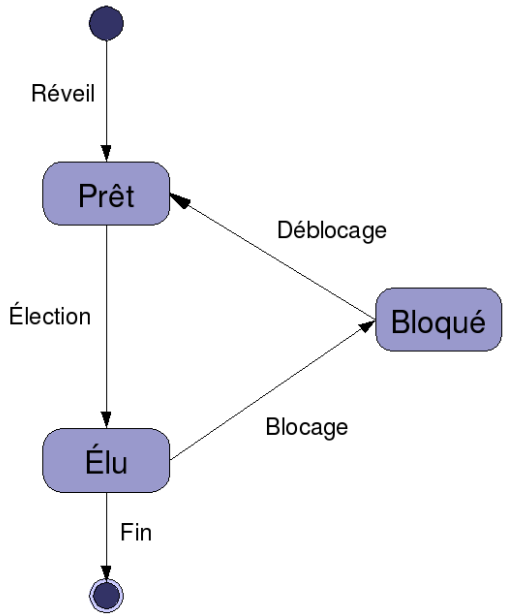
Un processus (lourd) est un executable en mémoire.

- Un processus lourd peut se dupliquer par un appel systeme appelé communement un fork.
- Lors d'un fork, toutes les pages mémoires alloués au père sont dupliqués et copiés dans son fils.
- Il n'est pas donc pas possible de partager la mémoire directement entre processus lourds (ils ne partagent pas le même espace d'adressage virtuel).
- Les processus fils peuvent à leurs tour se dupliquer.
- C'est une opération coutêuse en temps et en mémoire.

Lien avec le sytème d'exploitation

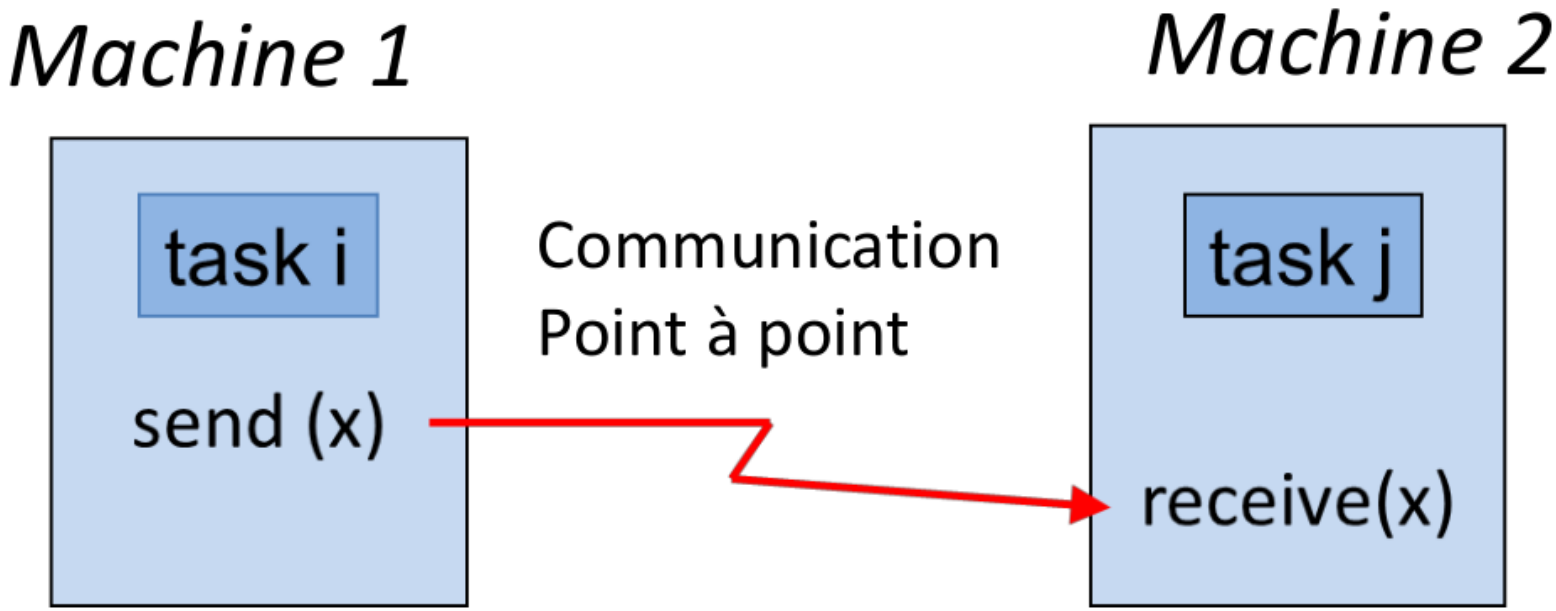
Le système d'exploitation du système :

- Alloue les ressources : mémoires, temps processeur, entrées/sorties nécessaires aux processus
- Garantit l'isolation : Assure que le fonctionnement d'un processus n'interfère pas avec celui des autres (zone mémoire virtuelle dédiée).
- Peut fournir une interface pour la communication inter-processus (comme la mémoire partagée sous linux).



Modèle par passage de messages

- Un ensemble de tâches travaillent sur leur mémoire locale (processus lourds) et s'éxecutent sur une ou plusieurs machines.
- Ces tâches échangent des données via des communication de messages.
- Un transfert de données est la résultante d'une coopération entre tâches.
- C'est au programmeur d'expliciter toutes les communications.
- Les messages peuvent être synchrones ou asynchrones.



Standard actuel : MPI (1993)

MPI (Message Passing Interface) est un standard définissant une bibliothèque de fonctions pour le passage de messages.

- Compatible C,C++ et Fortran
- Standard pour la communication inter-noeuds.
- Performance : Chaque fabricant optimise son implémentation MPI pour son matériel.
- Fournit également des mécanismes pour faire du RDMA (Remote Direct Memory Access).

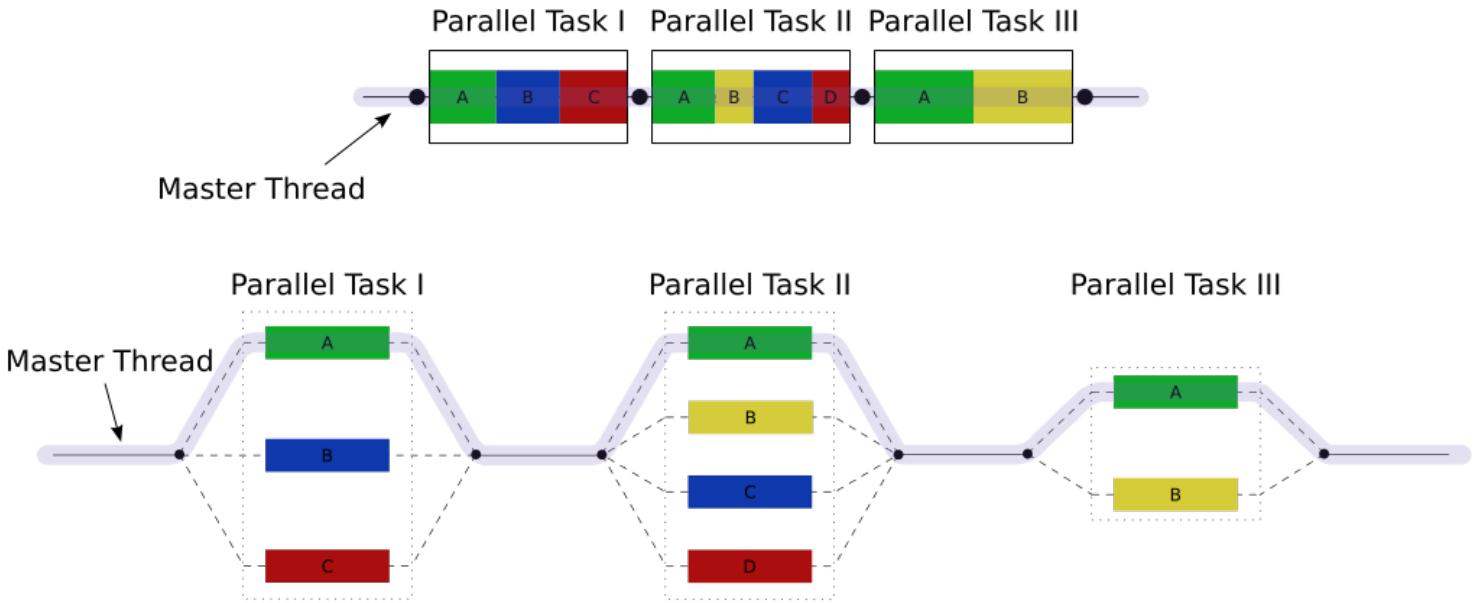
Les communications entre processus sont optimisées :

- Entre deux noeuds : Communication par le réseau.
- Entre deux coeurs d'un même noeud : Les implémentation choisissent en général le passage par la mémoire partagée.

Modèle des threads (processus légers)

Un processus peut initier plusieurs tâches concurrentes appelés threads :

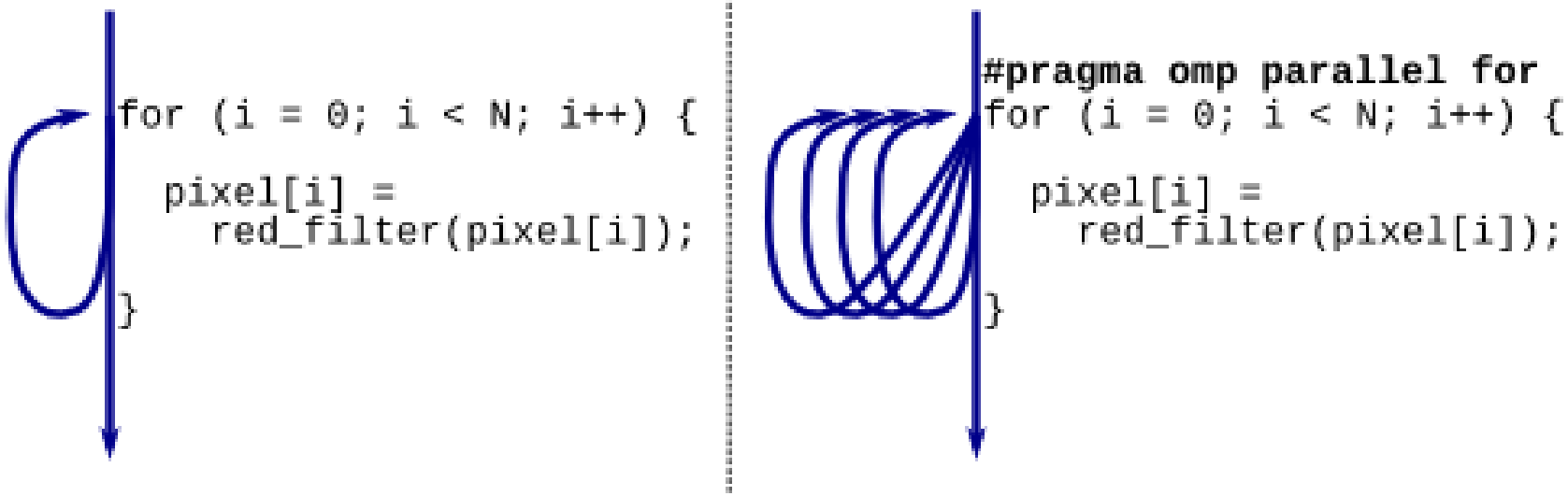
- Chaque thread possède sa mémoire locale et partage la mémoire globale du processus maître.
- La communication est donc possible via la mémoire globale.
- Un thread peut apparaitre, disparaitre puis réapparaitre (fork-join).
- Il faut synchroniser les accès concurrents (verrous, op. atomiques).
- Modèle d'execution des threads POSIX (Pthreads, 1995).



OpenMP (Open Multi-Processing, 1997)

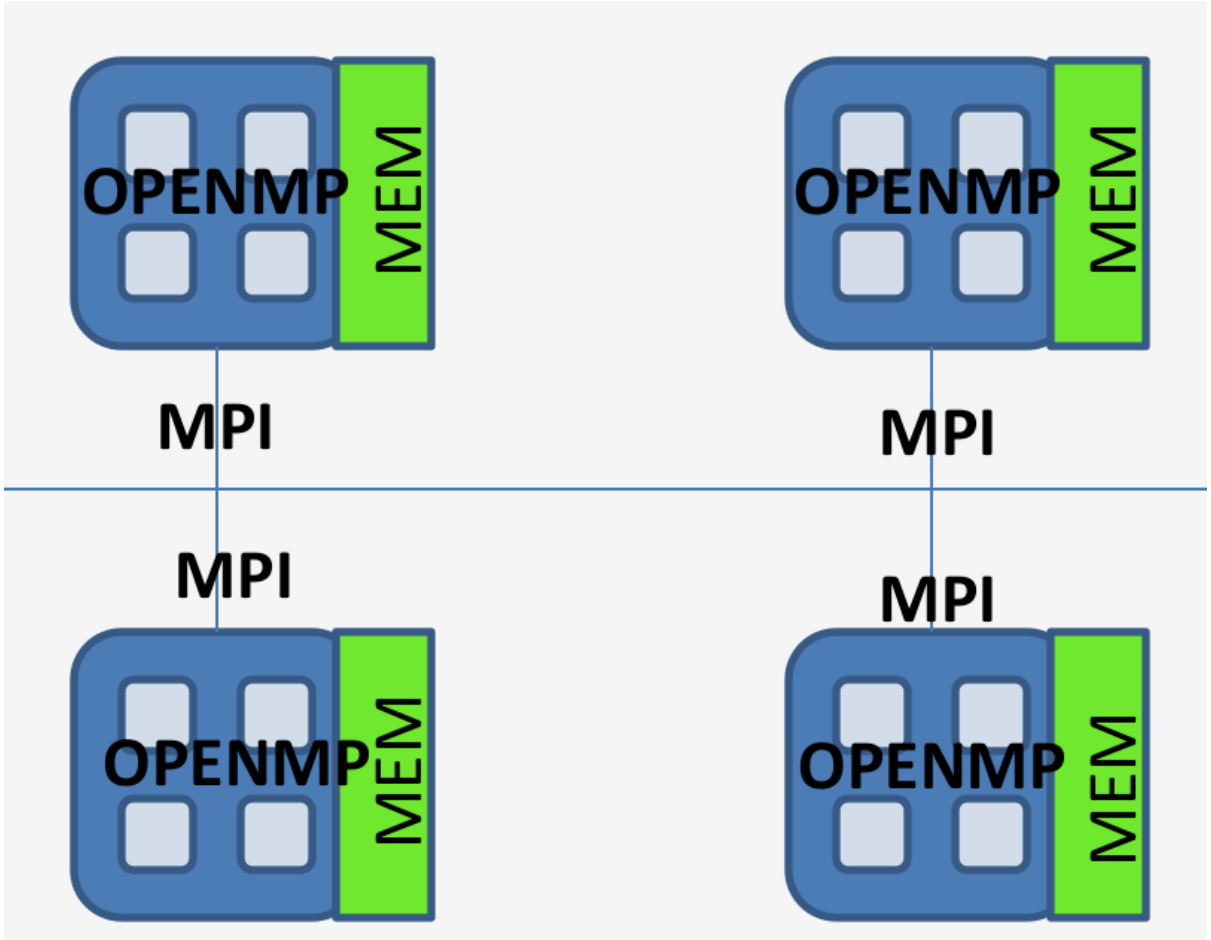
OpenMP est une interface de programmation pour le calcul parallèle sur architecture à mémoire partagée.

- Compatible C,C++ et Fortran.
- Adapté à la communication entre les noeuds d'une architecture à mémoire partagée.
- Utilisation de directives du compilateur.
- Permet de développer rapidement des application parallèles à petite granularité en restant proche du code séquentiel.



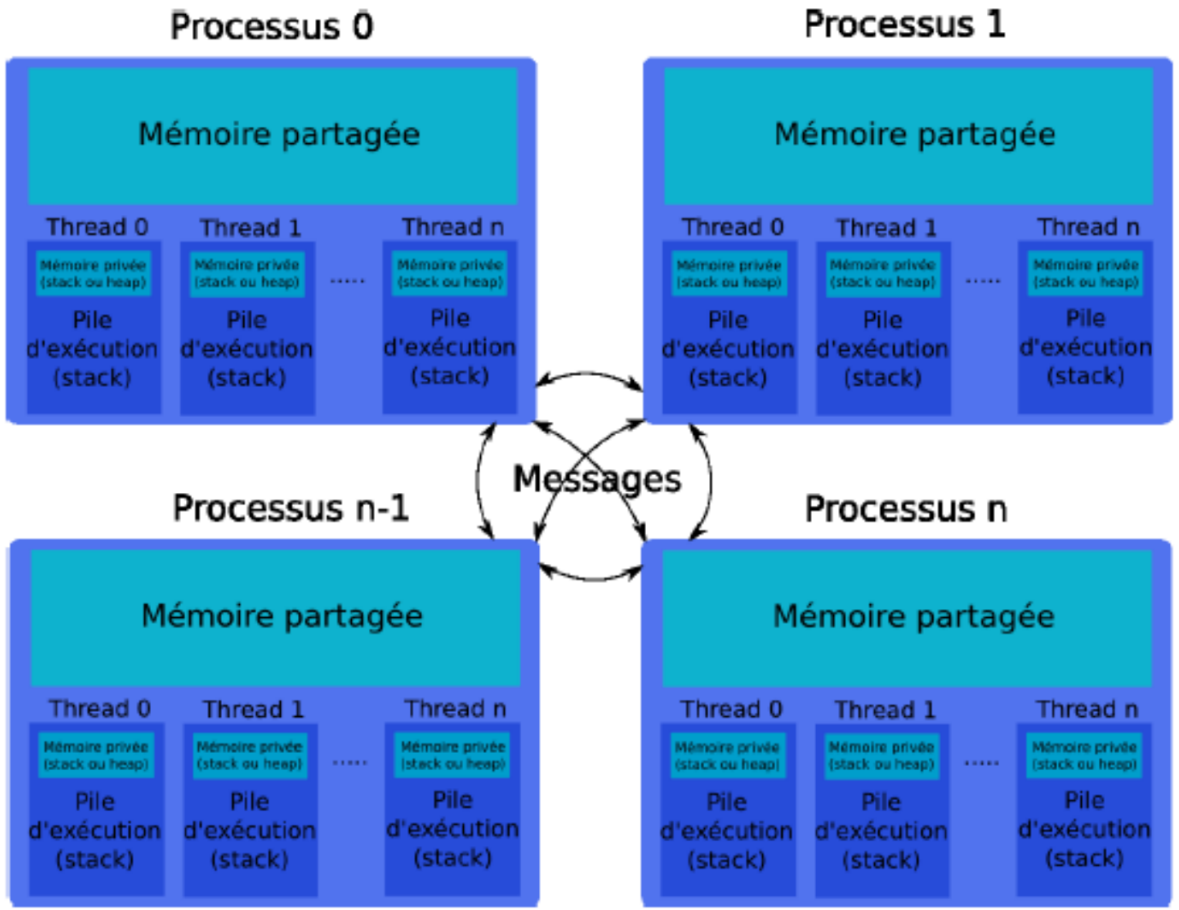
Parallélisme hybride

Le parallélisme hybride est basé sur l'utilisation combiné de plusieurs des modèles précédents.

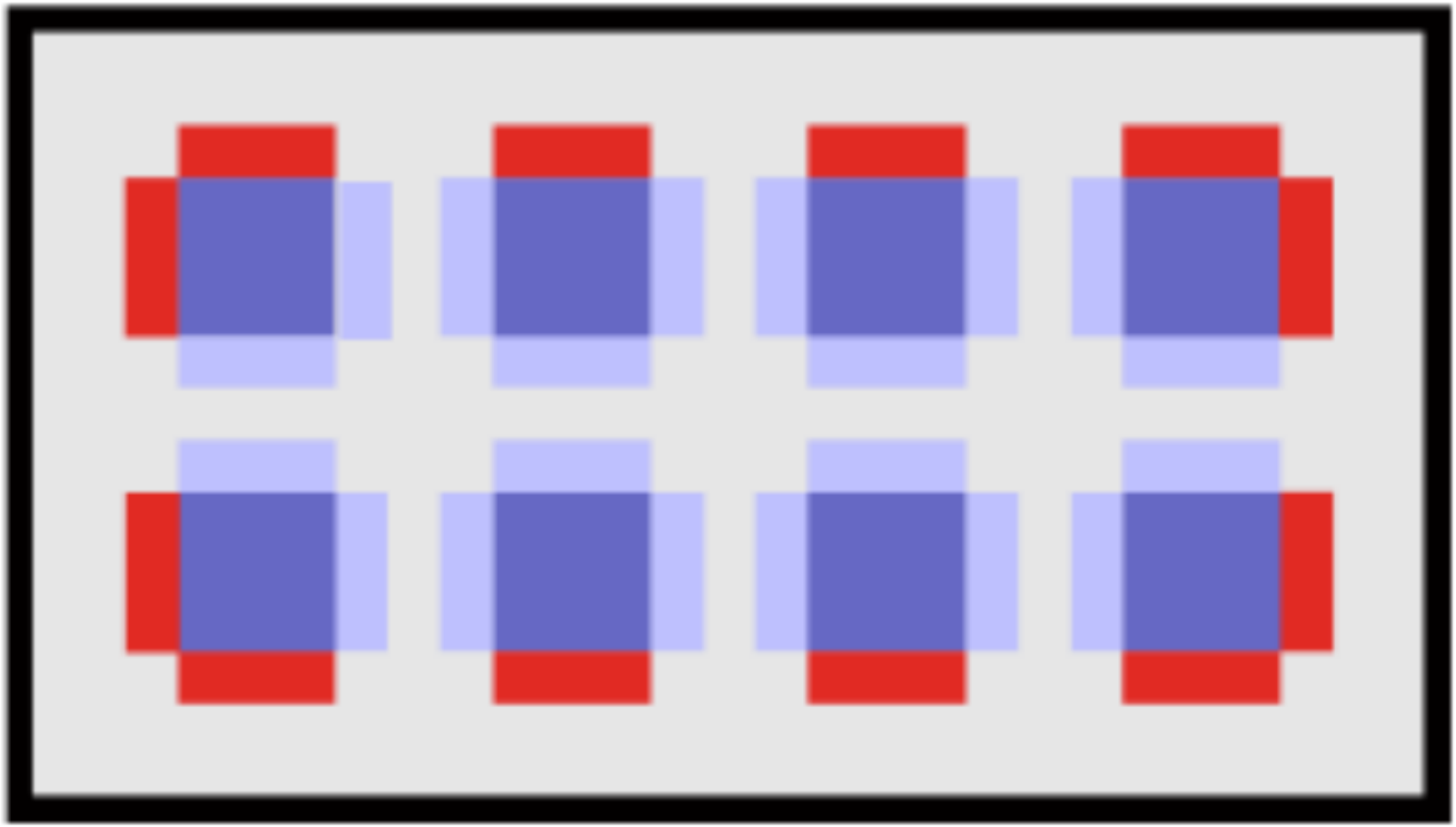


Parallélisme hybride

Le parallélisme hybride est basé sur l'utilisation combiné de plusieurs des modèles précédents.

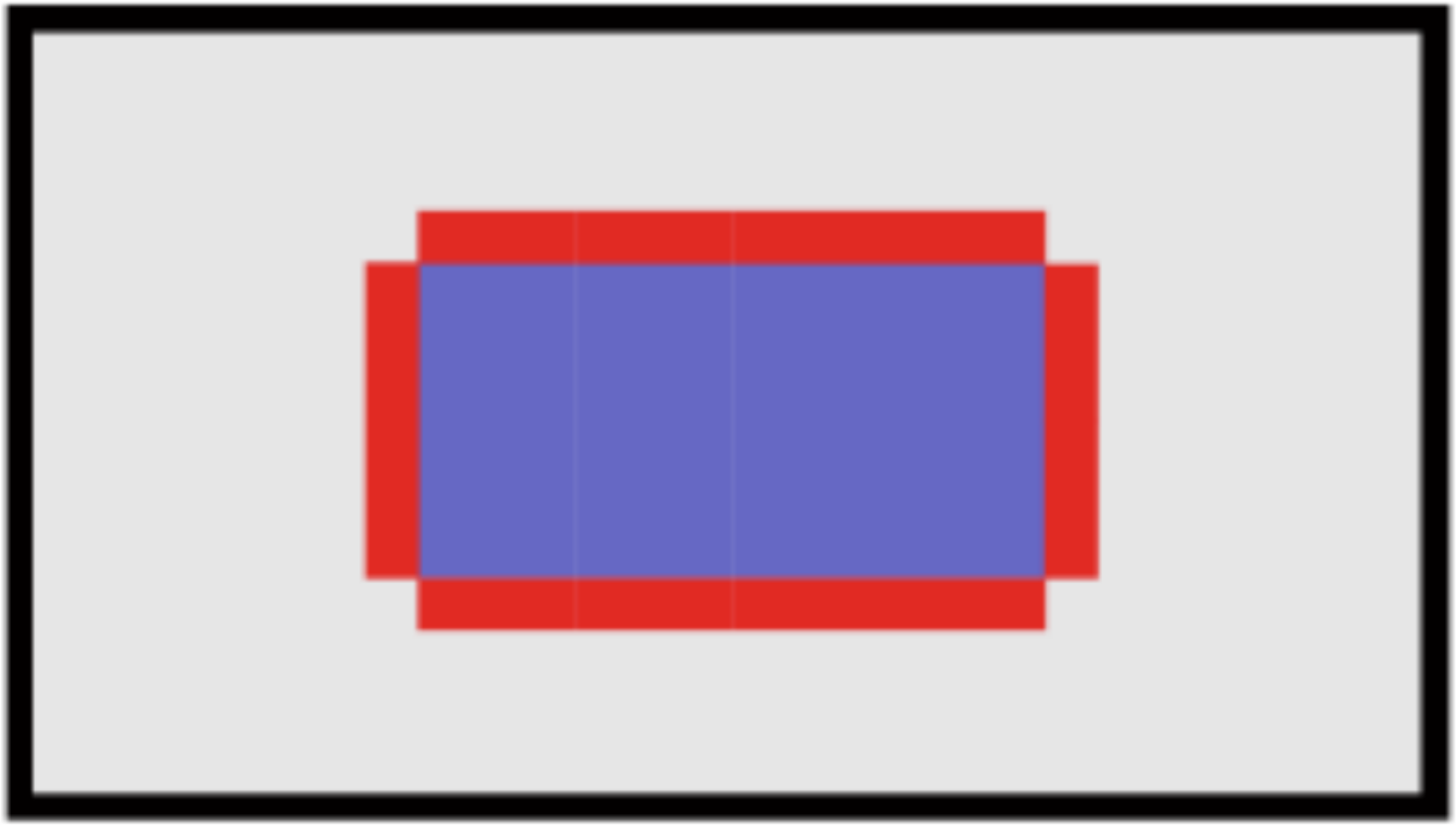


Parallélisme hybride : beaucoup de possibilités



Un processus MPI pour chaque coeurs de chaque noeud (8 coeurs ici)

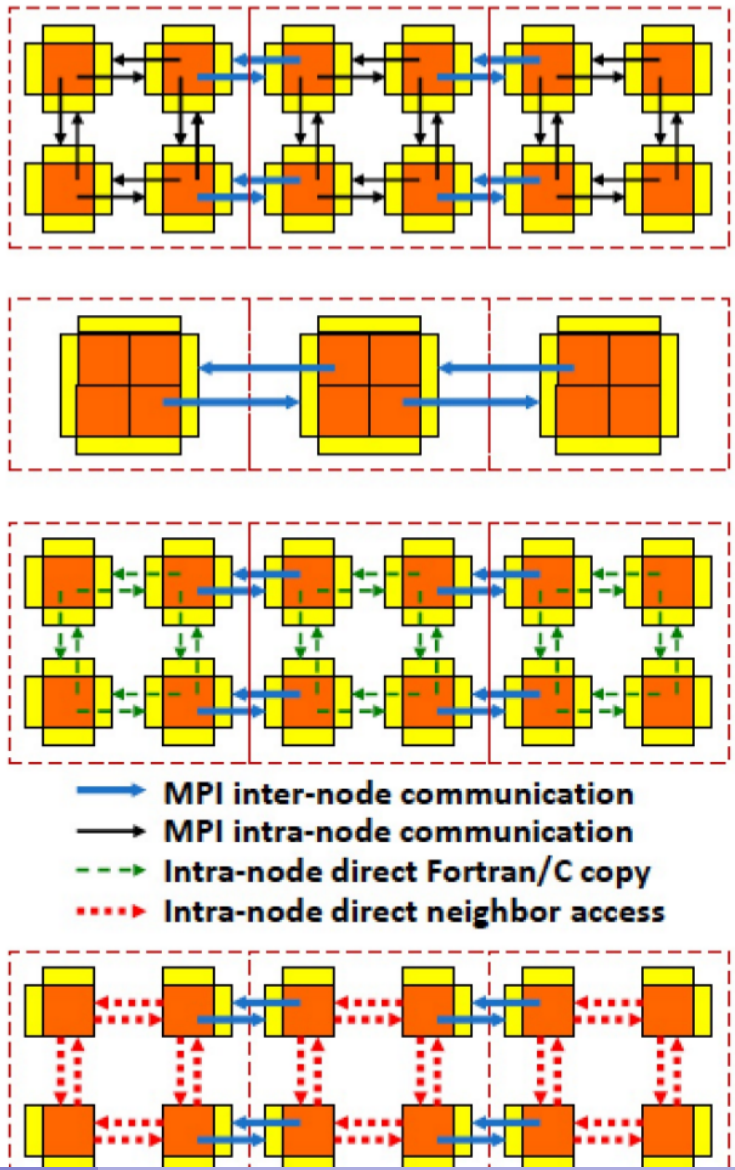
Parallélisme hybride : beaucoup de possibilités



Un seul processus MPI par noeud + mémoire partagée entre les coeurs d'un même socket (threads)

Parallélisme hybride : beaucoup de possibilités

- MPI on each core (not hybrid)
  - Halos between all cores
  - MPI uses internally shared memory and cluster communication protocols
- MPI+OpenMP
  - Multi-threaded MPI processes
  - Halos communica. only between MPI processes
- MPI cluster communication + MPI shared memory communication
  - Same as “MPI on each core”, but
  - within the shared memory nodes, halo communication through direct copying with C or Fortran statements
- MPI cluster comm. + MPI shared memory access
  - Similar to “MPI+OpenMP”, but
  - shared memory programming through work-sharing between the MPI processes within each SMP node



1 Architectures

2 Concepts du parallélisme

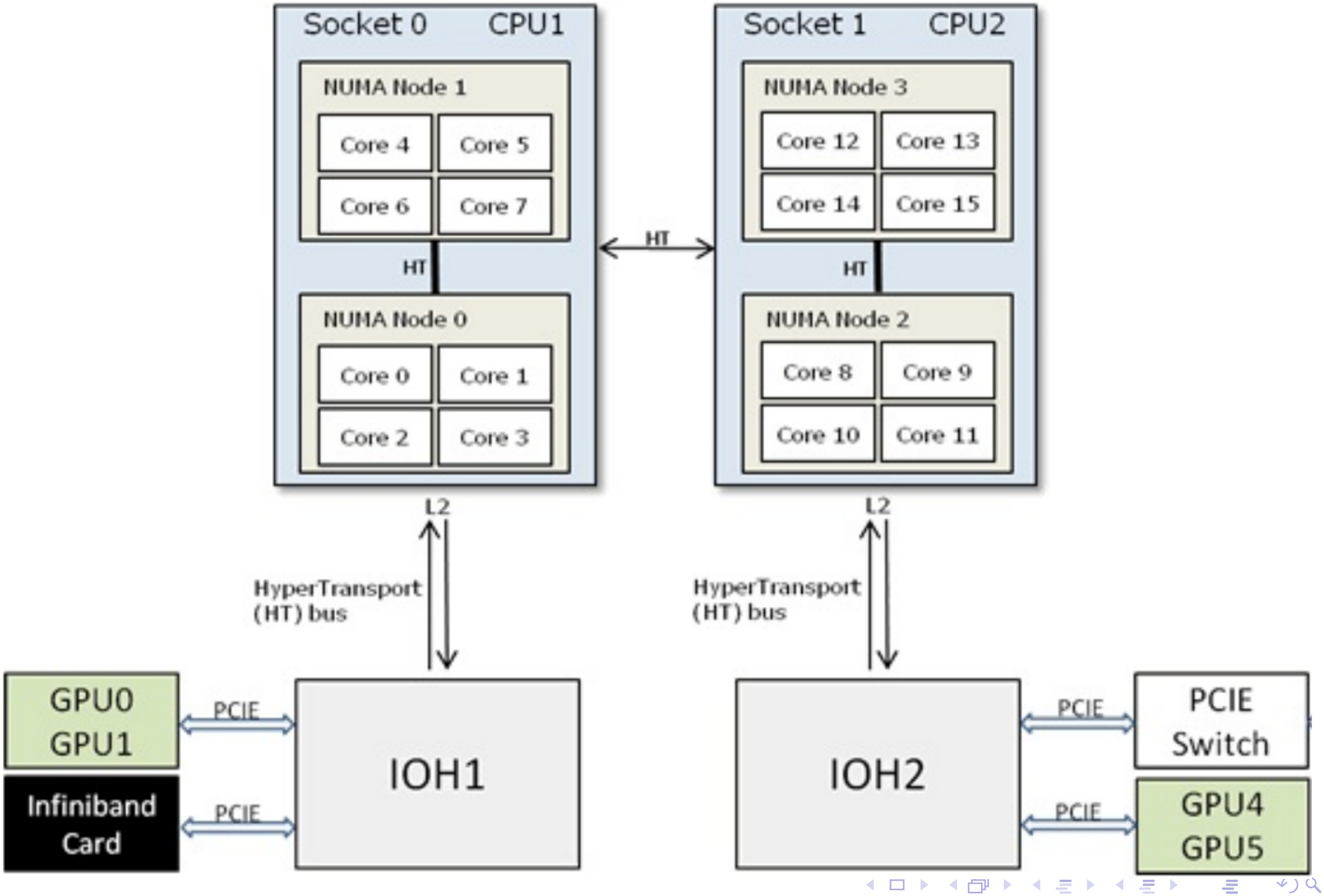
3 Les accélérateurs
- Motivations
- Inteface avec l'hôte
- Architectures
- Performances
- Parts des systèmes avec coprocesseur dans le TOP500
- Modèles de programmation pour coprocesseur

4 Conclusion

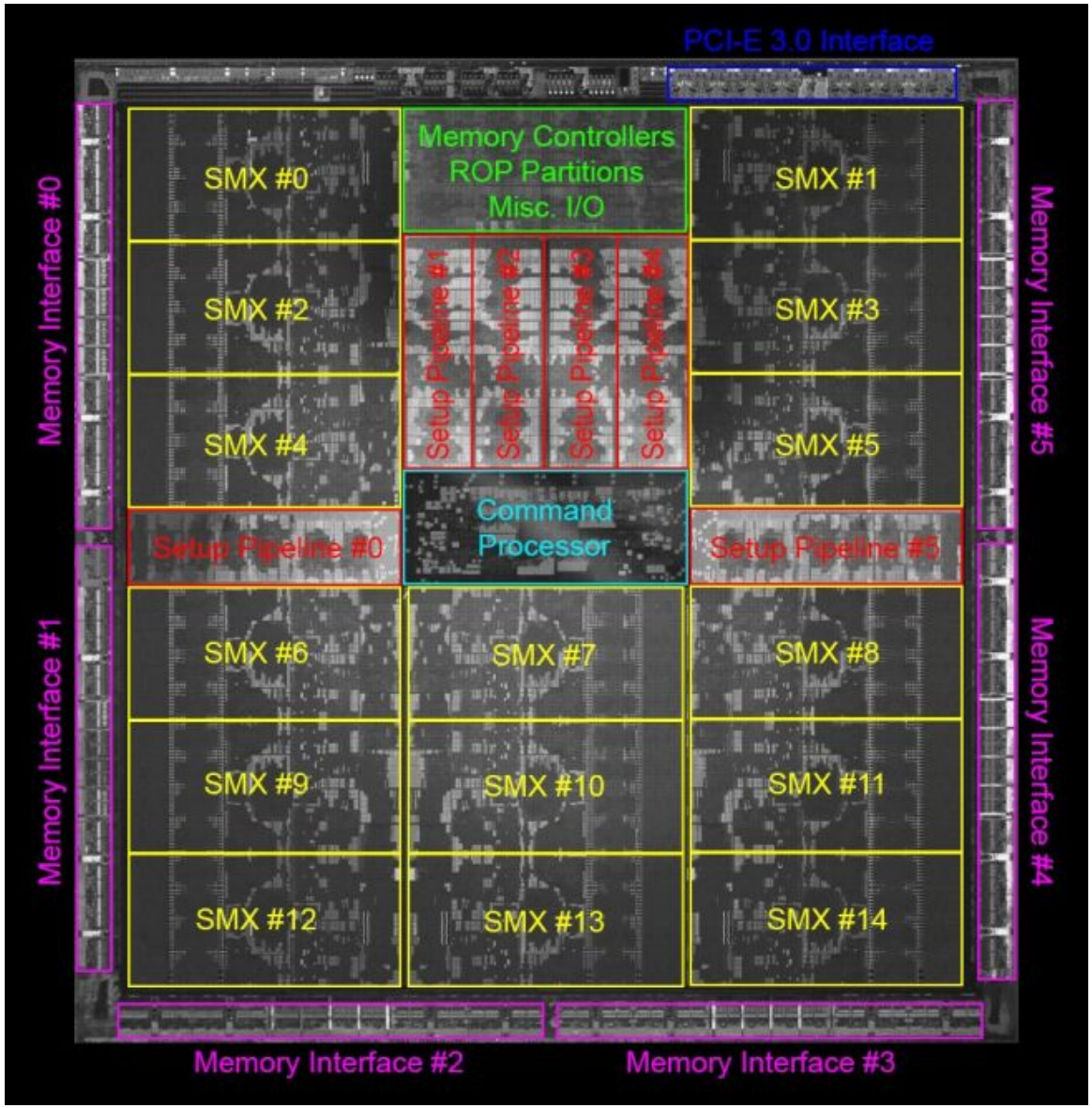
Motivations

- Certains codes numériques se prêtent bien au paradigme SIMD, où l'on applique un seul procédé un des nombreuses données en même temps.
- Les opération superscalaires sont limitées à 512bits sur un petit nombre de coeurs.
- Il est possible de créer des coeurs de calcul avec beaucoup moins de fonctionalités que des coeurs CPU, et qui s'éxécutent en même temps par groupes (ils ne sont plus indépendants).
- Historiquement ce type d'architecture nous vient surtout des cartes graphiques.
- Des modèles et des environements de programation de plus en plus mûres (développés depuis 2005).

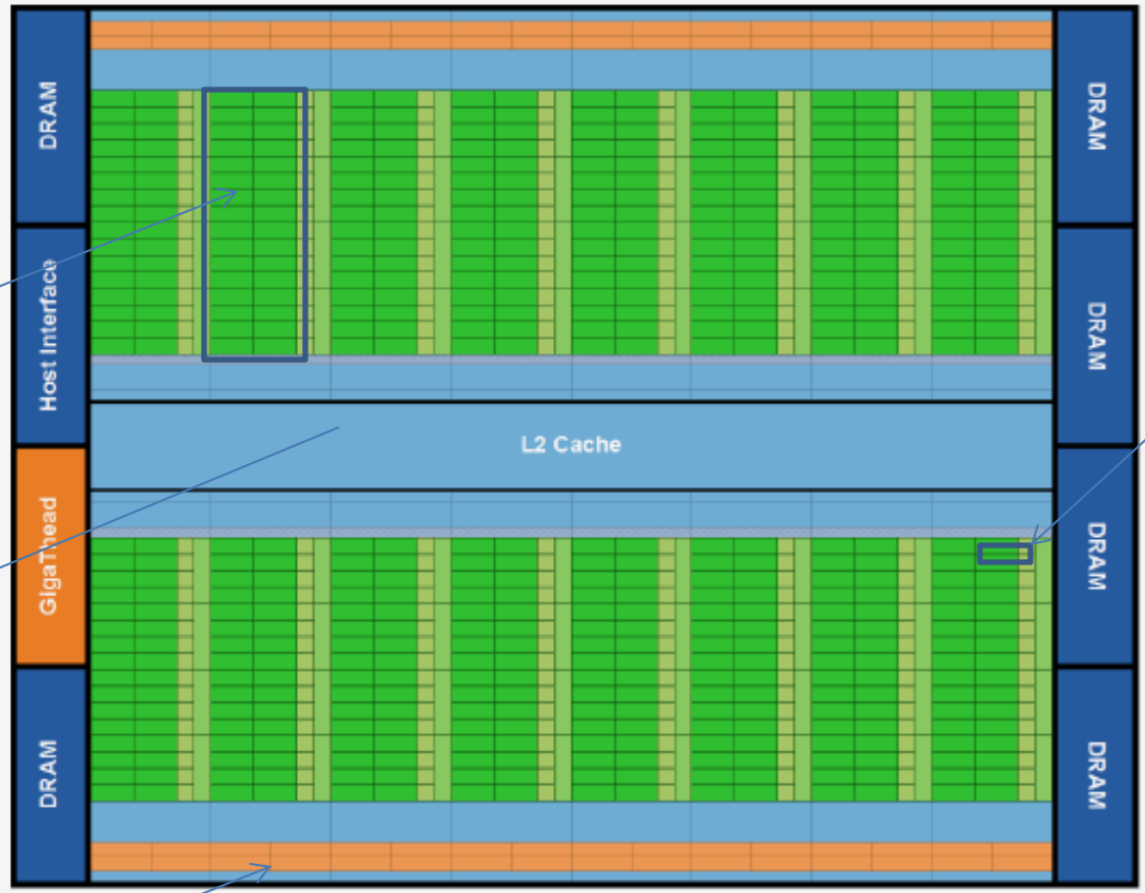
Intefaçage avec l'hôte



Architecture des GPUs



Architecture des GPUs



Multiprocesseur
SM

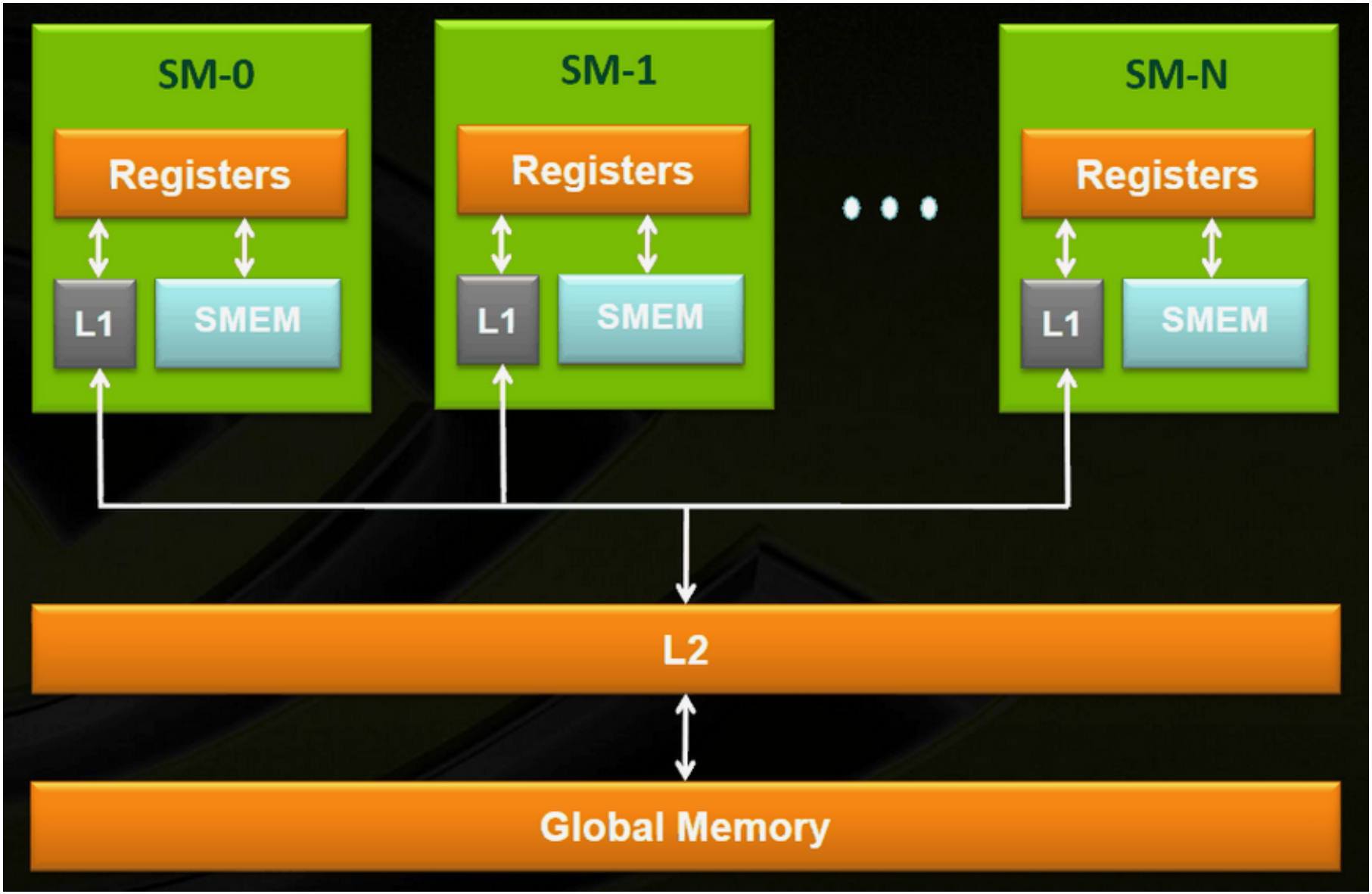
core

Cache L2 de 768 Ko

Shared memory / cache L1
64 Ko/SM

512 Unités de calcul !
16 partitions de 32

Architecture des GPUs

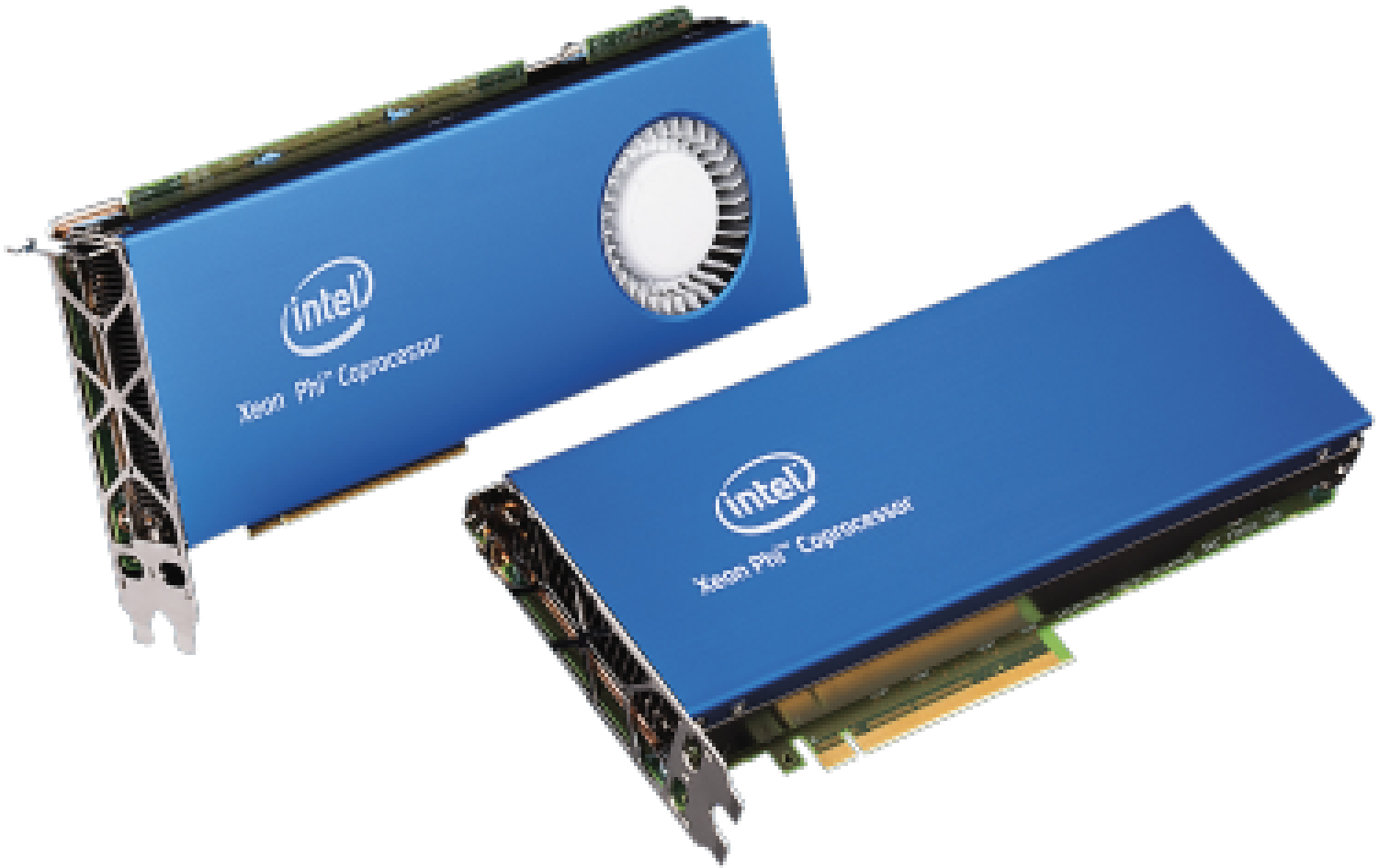


Les cartes concurentes :

Nvidia Tesla P100

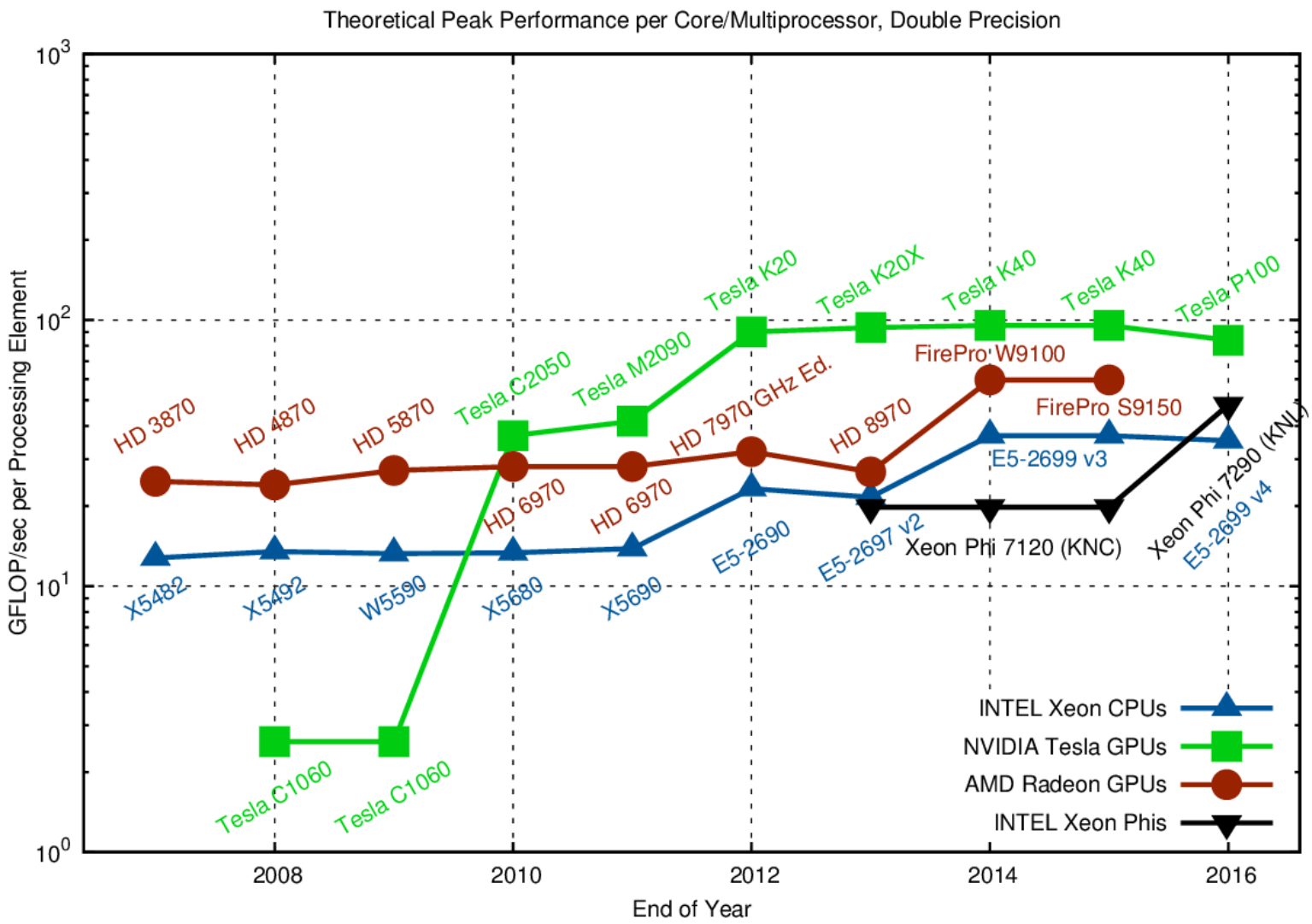
AMD Firepro W9100

La réponse d'intel :



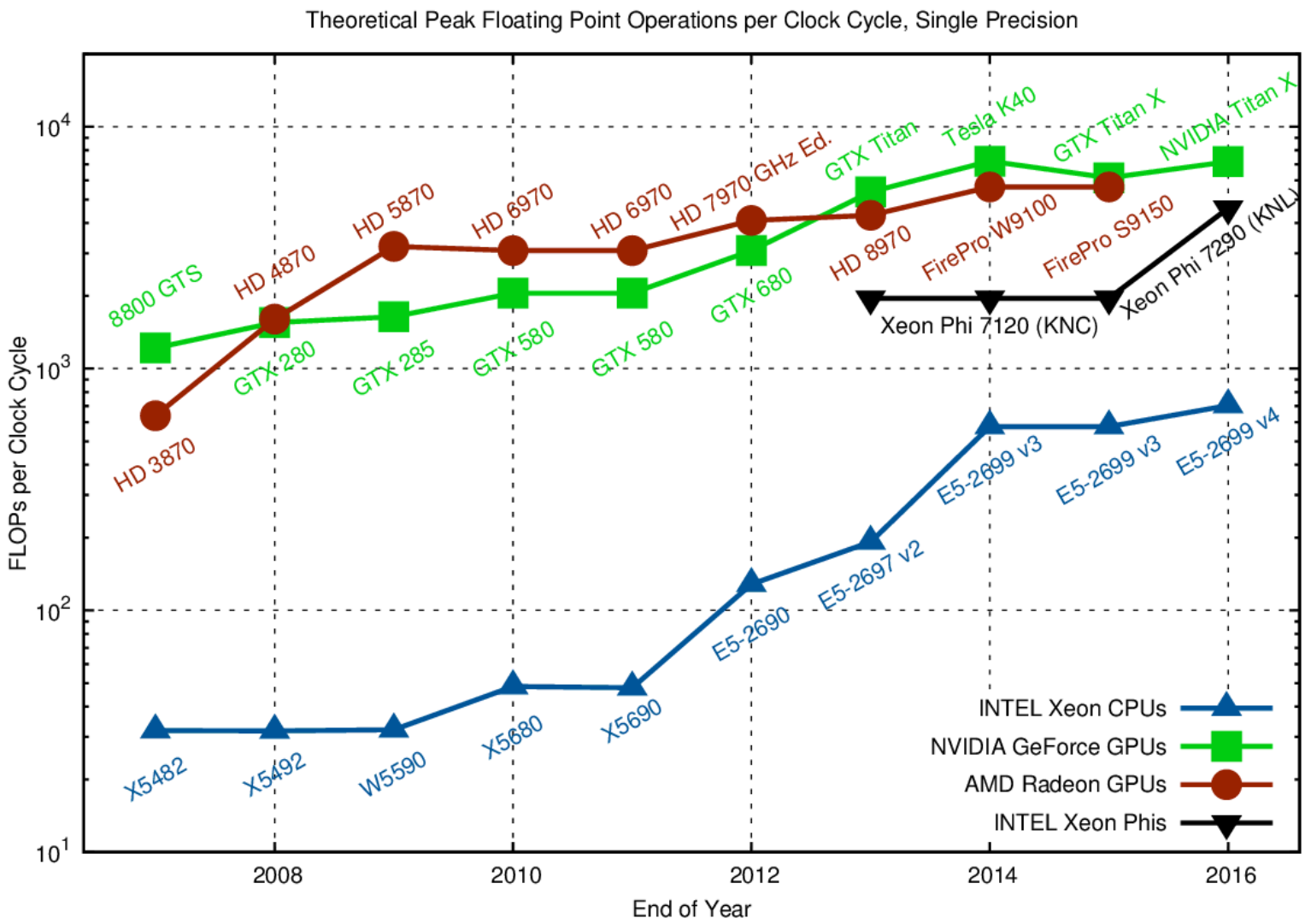
Coprocesseurs "manycores" type Xeon-Phi (72 coeurs à venir)

Performances



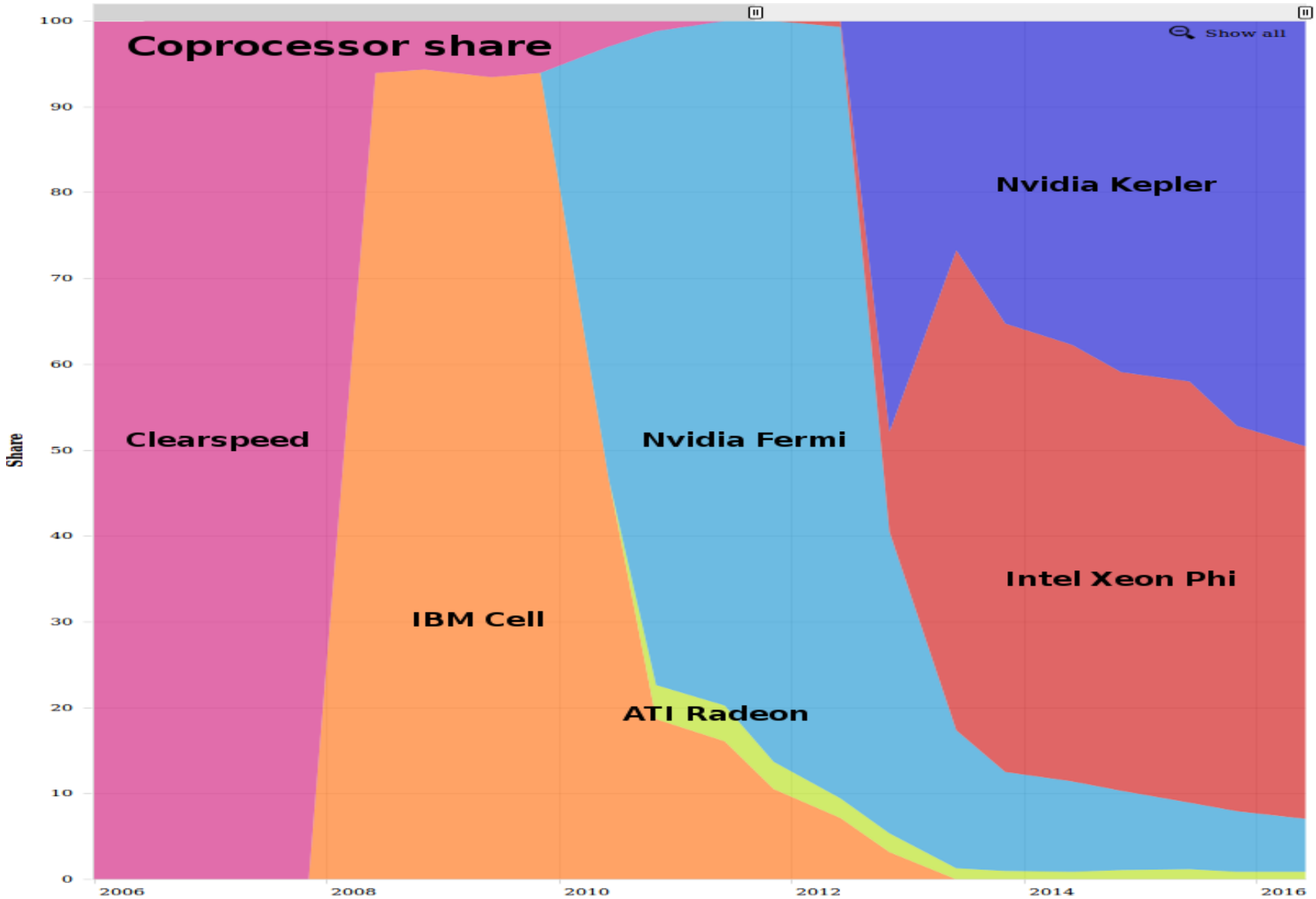
Performance théorique maximale en double précision par unité de calcul.

Performances



Performance théorique maximale en double précision.

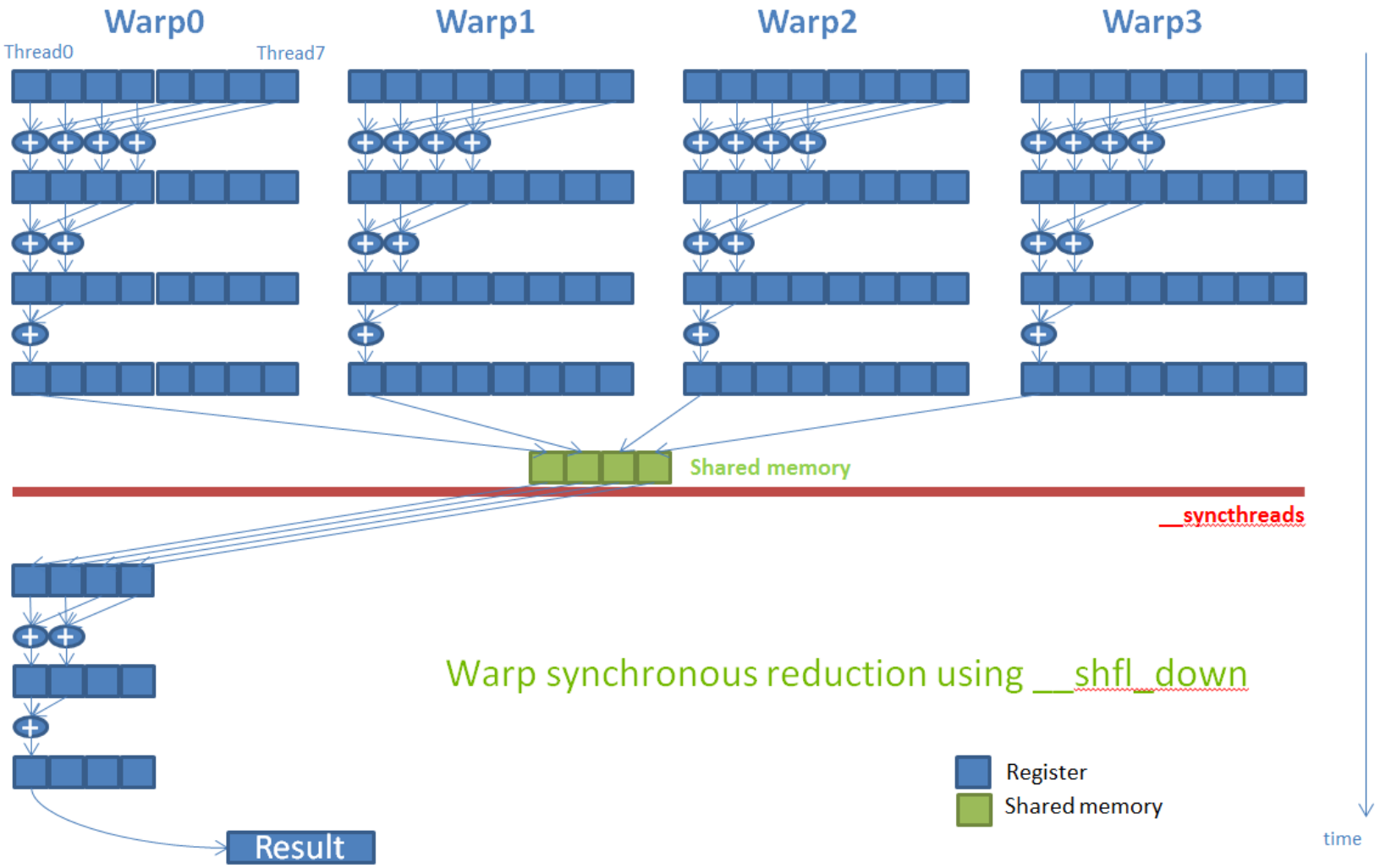
Parts des systèmes avec coprocesseur dans le TOP500



Les algorithmes qui marchent bien

Généralement : Données régulières, peu de chargement mémoire, beaucoup de calculs identiques et indépendants.

Exemple : algorithme de reduction (loi assosiative : max, +, ...)



Gains de performances sur des librairies de références

Gain d'un accélérateur Nvidia K40m face à un processeur Intel E5-2697 12coeurs@2.70GHz :

- cuBLAS vs mkIBLAS (Basic Linear Algebra Subroutine) : 6-17x
- cuSPARSE vs mkISPARSE (Sparse Matrix Library) : 5x
- cuRAND vs mkIRAND (Random Number Generation) : 70x
- cuFFT vs mkIFFT (Fast Fourier Transform) : 2-10x (10-50x sans copie)

Modèles de programmation pour coprocesseur

- OpenCL (Open Computing Language) : Permet l'éxecution de code à la fois sur CPU et GPU. Cela n'est pas synonyme de portabilité des performances. Basé sur un sous ensemble du C99 (2008).
- CUDA (Compute Unified Device Architecture) : Equivalent propriétaire d'OpenCL pour les GPUs NVidia. Basé sur un sous ensemble du C99 et étendu à un sous ensemble du C++14 (2007).
- OpenACC (Open Accelerators) : Comme OpenMP, ajout de commandes préprocesseur directement dans le code source (2012).
- OpenMP (Open Multi-Processing) : depuis sa version 4.0 (2013).
- Tous les modèles hybrides précédent avec ces différentes variantes...

Conclusion

- Vérifier d'abord si un code parallèle n'est pas déjà disponible.
- Identifier les parties coûteuses en temps de calcul (profiling).
- Commencer par améliorer les performances du code séquentiel afin de se focaliser ensuite uniquement sur la parallélisation du code.
- Identifier les opérations qui interdisent la parallélisation (dépendance de données).
- Rechercher des algorithmes éventuellement mieux adaptés à la parallélisation.
- Tenir compte de l'architecture cible (nb de noeuds, hiérarchie mémoire, présence d'accélérateurs ?).
- Malheuresement les architectures évoluent très vite et il faut adapter continuellement les codes.

Références

- Les séminaires de Françoise Roch (mésocentre CIMENT).
- `www.idris.fr` séminaires HPC de l'IDRIS
- `www.citutor.org` ressources HPC (MPI, OpenMP, outils, librairies, ...)
- `www.top500.org` le TOP500 (actualisé tous les 6 mois)
- `www.gpgpu.org` actualité et séminaires les technologies GPU