Demandez le Programme !

Optimisations matérielles pour les architectures actuelles
pour dépasser les limitations des performances séquentielles

Fin de la loi de Moore

- On continue à graver des transistors plus fins, l'écart entre deux générations de gravure n'est plus de 18 à 24 mois, mais plutôt deux fois plus.

- Des transistors plus fins ne veulent plus dire une puce plus petite et donc moins chère, car l'écart entre deux transistors ne décroit plus avec la finesse à cause des interférences électriques et des problèmes de fabrication.

- Le procédé de fabrication influence la topologie des composants : les mémoires et processeurs simples continuent à passer à l’échelle, mais pas les schémas optimisés et complexes.

Mur de la chaleur

- La consommation électrique croit avec le carré de la fréquence, et la surface d'un processeur décroit avec le carré de la finesse de gravure. La puissance totale en Watts est constante, mais la puissance par cm^2 tend vers l'infini !

- Afin de gagner en vitesse, il faut raccourcir les distances, on empile donc les composants en 3D, ce qui pose une contrainte d’extraction de chaleur encore plus forte.

Mur de la mémoire

- Le temps d'accès stagne à 60ns (=16 MHz) et il est inutile que le processeur aille plus vite que quelques GHz, car un défaut de cache coûte alors plusieurs centaines de cycles-machine et la plupart des algorithmes ne font qu'attendre la mémoire.

- La solution apportée est l'élargissement du bus, ce qui revient à de la parallélisation : DDR4 = 1024 bits.

- Un 3e niveau de hiérarchie mémoire est apparu : la HBM se situe entre le cache et la DRAM externe.

- Et de 4 ? : Intel 3DXPoint et HP Memristor

Limitations du multicœur

- Il faudrait idéalement autant de bus mémoire que de cœurs, mais la matrice d'interconnexion croît avec le nombre de bus multiplié par le nombre de cœurs, $O(n^2)$. C'est pourquoi le nombre de bus ne suit pas et stagne à 8 aujourd'hui.

- Augmenter le nombre de puces sur une même carte-mère entraine un ralentissement des accès mémoire (ccNUMA) et l’explosion du nombre de synchronisations de lignes de cache en $O(n^2)$.

Thin cores *vs* fat cores

- Limitation du procédé electro chimique, les composants simples comme la SDRAM, DRAM, SRAM ou processeurs simples et massivement parallèles (GPU, FPGU, ARM) continuent à bénéficier de meilleure finesse de gravure, mais c'est de plus en plus difficile pour les composants très optimisés et complexes comme l'architecture X86.

- On est aujourd’hui contraint d’optimiser la performance séquentielle des algorithmes et de les paralléliser afin de compenser la trop faible évolution de vitesse des cœurs.

Optimisation du nombre de calculs par Hz

- OOE : Out of Order Execution.
- RR : Register Renaming.
- SS : Super Scalar.
- Limite la capacité du composant à être gravé finement et utilise beaucoup de transistors.

Exécution spéculative

- Permets de ne pas effectuer un ensemble de calcul en déduisant par avance le résultat grâce à un cache de résultats précédents. Permets d’aller plus vite que la lumière !

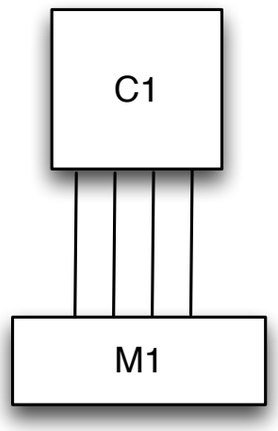
- Danger : permets aussi au hacker de trouver des failles de sécurités (Specter et Meltdown).

- Un point de rencontre intéressant entre les théories mathématiques les plus «fumeuses» et l’accélération concrète des processeurs.

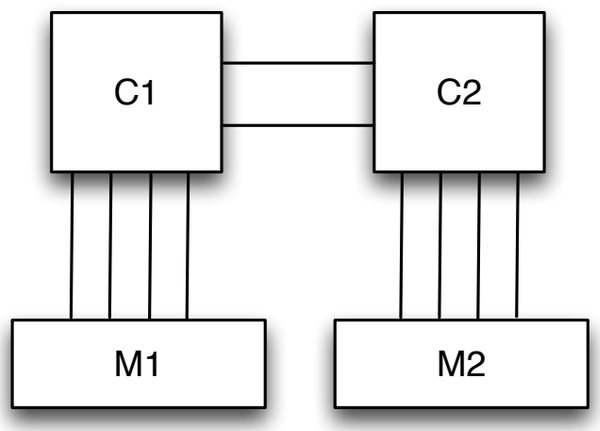
DSHM et NUMA

- DSHM : Distributed Shared Memory
- NUMA : Non Uniform Memory Access
- Utilisation de plusieurs puces, chacune gérant une portion de la mémoire. La mémoire de chaque puce est accessible de manière transparente par les autres via un canal dédié (QPI). C’est efficace avec deux puces, mais au-delà il faut impérativement tenir compte du placement des processus et de la mémoire (numactl).

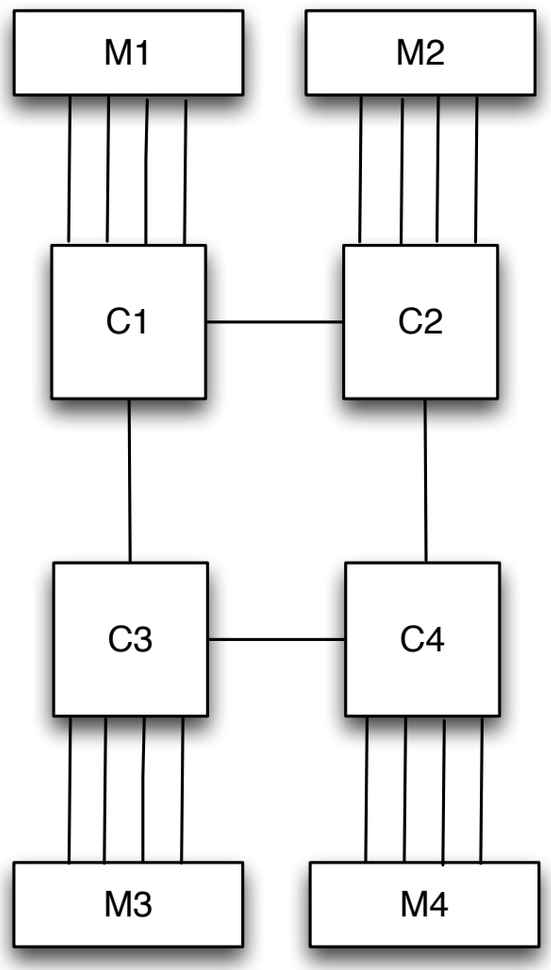
DSHM et NUMA



1 CPU:
latency = 100 ns
throughput 60 GB/s



2 CPUs:
latency = (100 + 200) / 2 = 150 ns
throughput (60 + 30) / 2 = 45 GB/s



4 CPUs:
latency = (100 + 200 + 200 + 300) / 4 = 200 ns
throughput (60 + 10 + 10 +10) / 4 = 22.5 GB/s

Unitées vectorielles

- La complexité croit linéairement avec la taille des vecteurs.
- Ces dernières années, la vectorisation compense la faiblesse du multicœur.
- $N_c * T_v * F_{GHz}$ = doublement tous les deux ans.
- Vectorisation explicite : sur CPU, beaucoup de travail.
- Vectorisation implicite : sur GPU, démarrage long, mais plus simple sur le long terme.