

# Modèles et techniques de programmation parallèle hybride et multi-cœurs

Introduction, rappels sur l'architecture matérielle, différents types de parallélismes

Marc Tajchman

CEA - DEN/DM2S/STMF/LDEI

31/08/2025

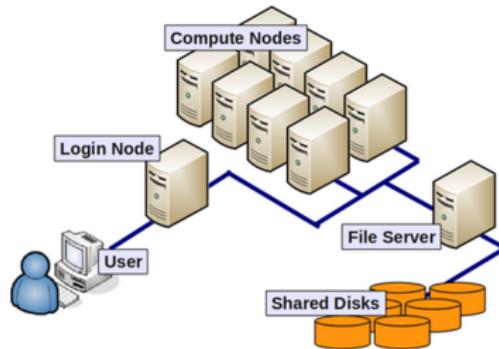
# Rappels sur l'architecture matérielle

La plupart des ordinateurs actuels ont des fonctionnalités parallèles (exécution simultanée de plusieurs traitements) :

- ▶ Les ordinateurs personnels (ordinateurs portables ou de bureau, téléphones portables, tablettes, etc.) ont presque toujours plusieurs unités de calcul (ou cœurs).  
Dans chaque cœur, il y a souvent plusieurs circuits arithmétiques (additionneurs, moltiplicateurs, etc).
- ▶ Beaucoup d'ordinateurs ont, de plus, des dispositifs spécialisés dans le calcul (exemple: (GP)GPU, carte graphique utilisée pour du calcul "massivement parallèle")
- ▶ Les machines parallèles regroupent plusieurs nœuds de calcul (chaque nœud de calcul est similaire à un ordinateur personnel, sans clavier, sans écran, ni souris), les nœuds de calculs sont connectés entre eux.

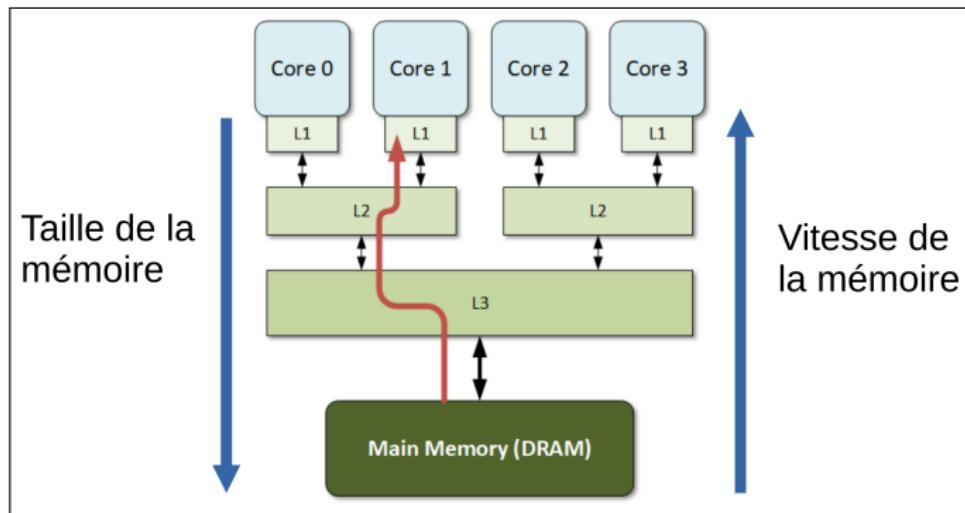
## Machine parallèle typique:

- ▶ un ensemble de nœuds de calcul chacun contenant un ou plusieurs cœurs et de la mémoire, parfois un GPU
- ▶ les nœuds sont connectés entre eux par un réseau
- ▶ et aussi à un système qui gère les données d'entrée et produites (disques partagés)
- ▶ les utilisateurs ne se connectent pas directement aux nœuds de calcul mais passent par un frontal (ordinateur qui gère les connexions multiples à la machine parallèle)



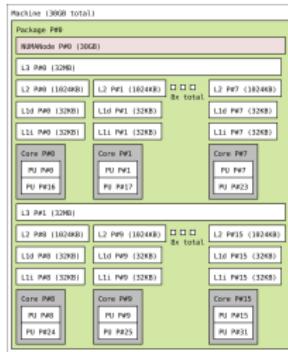
(les machines les plus puissantes actuellement contiennent plusieurs centaines de milliers de nœuds)

Schéma interne (très simplifié) d'un nœud (variable suivant le modèle de processeur et la génération utilisée):

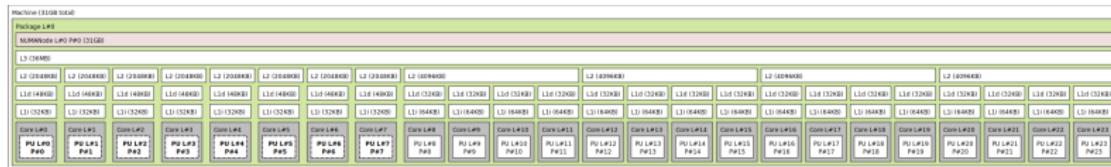


Chemin rouge: trajet d'une donnée entre la mémoire centrale et un des cœurs dans le processeur.

## Exemple : processeur AMD Ryzen9 7950X



## Exemple : processeur Intel i9 13980HX



Jusque 6 types de mémoires différentes:

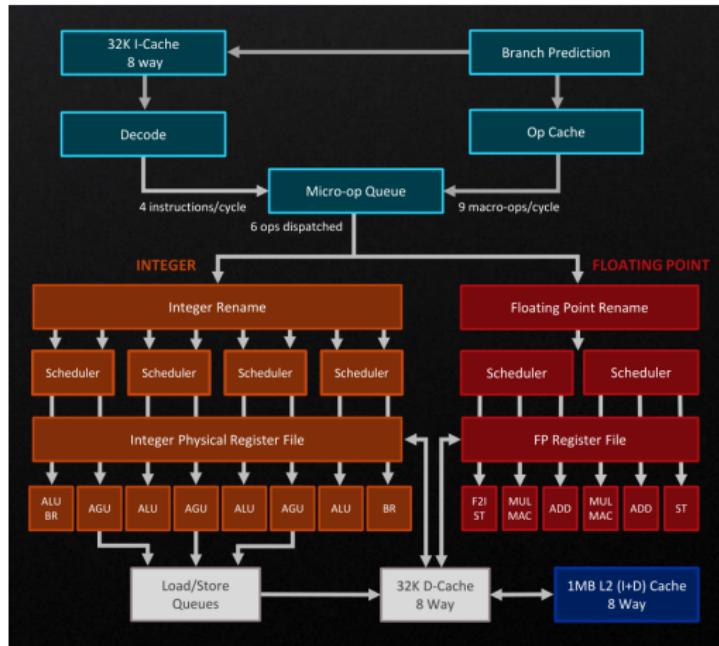
- ▶ registres : mémoire très rapide dans chaque cœur
- ▶ mémoires caches L1, L2, L3 : mémoires rapides, partagées entre une partie des cœurs
- ▶ mémoire centrale : mémoire moins rapide mais de plus grande taille
- ▶ disques : mémoire lente, mais de très grande taille.

A noter que la différence entre mémoire centrale et disque tend à diminuer avec la généralisation des SSD.

Les échanges entre la mémoire centrale et les mémoires cache se font par blocs de données (appelées "lignes de cache") et pas "à l'unité" pour des raisons de performances.

*Dans la plupart des machines, une ligne de cache contient 64 octets.*

# Schéma interne d'un cœur (variable suivant le modèle de processeur et la génération utilisée):



AMD - Zen4 (simplifié)

Le cœur contient en général plusieurs unités de traitement pour les opérations sur les entiers et les nombres décimaux (addition, multiplication, etc.)

Situation actuelle (et encore pour plusieurs années):

*vitesse de calcul (d'un processeur)*

$\approx$  *vitesse de la mémoire interne (registres) du processeur*

> *vitesse des différentes mémoires cache, intermédiaires entre le processeur et la mémoire centrale du nœud*

(vitesse  $L1 > L2 > L3$ ,  
taille  $L1 < L2 < L3$ )

$\gg$  *vitesse de la mémoire centrale d'un nœud*

$\gg$  *vitesse du réseau qui connecte les nœuds*

La gestion des différentes mémoires est faite par une puce appelée contrôleur mémoire:

- ▶ Quand un cœur a besoin d'une donnée, il la demande au contrôleur mémoire.
- ▶ Le contrôleur mémoire regarde si la donnée est dans la mémoire cache L1.
- ▶ Si la donnée n'est pas dans la mémoire L1, le contrôleur mémoire regarde si la donnée se trouve dans la mémoire cache L2, dans la mémoire L3 ou dans la mémoire centrale. Le bloc mémoire qui contient la donnée est remonté dans la hiérarchie de cache jusqu'à L1.
- ▶ Finalement, la donnée est copiée de L1 dans un registre de ce cœur.

- ▶ Quand un cœur a modifié une donnée, il le notifie au contrôleur mémoire
- ▶ Le contrôleur mémoire recopie la mémoire vers le cache L1, redescend le bloc donnée vers la mémoire cache (L2 puis L3) et vers la mémoire centrale
- ▶ Il vérifie aussi qu'une autre copie de la donnée n'existe pas dans la mémoire cache d'un autre cœur.
- ▶ Dans ce cas, les autres copies sont mises à jour

Ce système s'est généralisé parce qu'il est très compliqué/cher de produire de la mémoire qui suive les performances des processeurs.

Permet de maintenir de bonnes performances avec de la mémoire de grande taille mais plus lente.

**Cette gestion peut représenter une partie importante du temps d'exécution de programmes parallèles (pour assurer la cohérence des différentes parties de la mémoire et leurs mises à jour correctes).**

## Ce qu'il faut retenir:

- ▶ En général, plusieurs niveaux de parallélisme disponibles
  - ▶ Entre les nœuds de calcul (avec par ex. MPI)
  - ▶ Entre les cœurs dans un même nœud de calcul (avec par ex. MPI ou OpenMP)
  - ▶ Dans un cœur, entre les circuits arithmétiques (par ex. avec les instructions SSE, AVX)
  - ▶ Dans un GPU (avec par ex. Cuda/OpenCL)
- ▶ On peut combiner ces différents niveaux de // (en faisant un peu attention)
- ▶ Dans ce cours, on s'intéressera principalement:
  - ▶ au // entre cœurs (OpenMP)
  - ▶ au // dans un GPU
  - ▶ au // mixte (MPI/OpenMP/Cuda)
- ▶ Tous les ordinateurs actuels utilisent de la mémoire cache  
Améliore le temps moyen d'accès aux données par le processeur, mais complexifie la gestion mémoire.