Modèles et techniques en programmation parallèle hybride et multi-cœurs

Marc Tajchman

CEA - DEN/DM2S/STMF/LMES

November 29, 2017

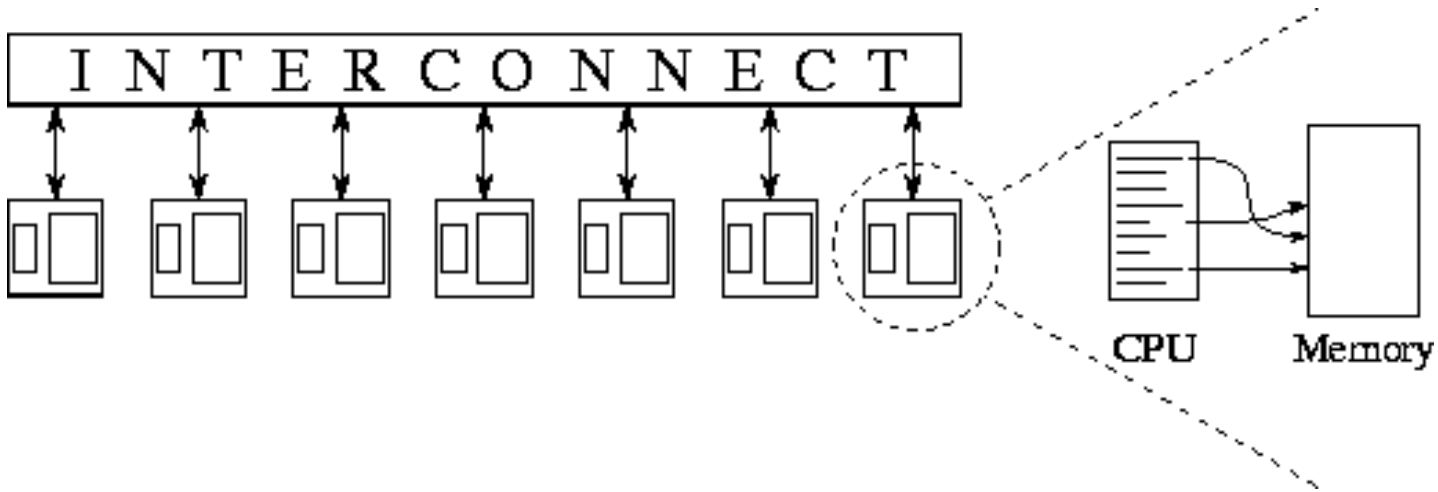
Plan

- Rappels sur l'architecture matérielle
- Optimisation de la programmation séquentielle
- Rappels de programmation parallèle
  - Rappel des notions
  - Mémoire partagée
  - Mémoire distribuée
- Programmation parallèle hybride
- Examen

Rappels sur l'architecture matérielle

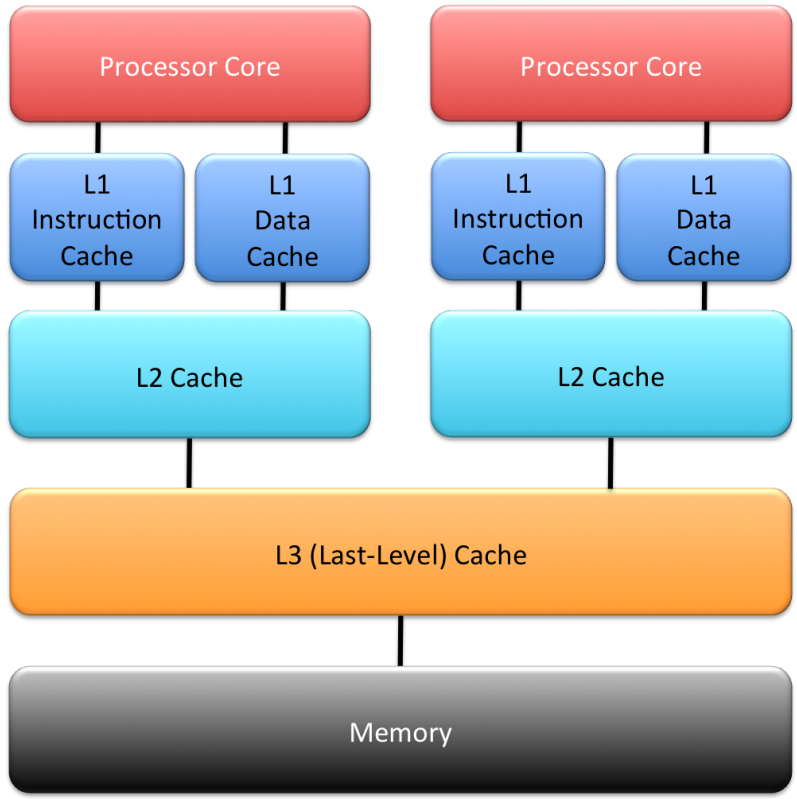
On peut voir la structure d'une machine de calcul avec différents grossissements:

Vue globale: un ensemble de nœuds de calcul chacun contenant un ou plusieurs processeurs et de la mémoire, les nœuds sont connecté par un réseau:

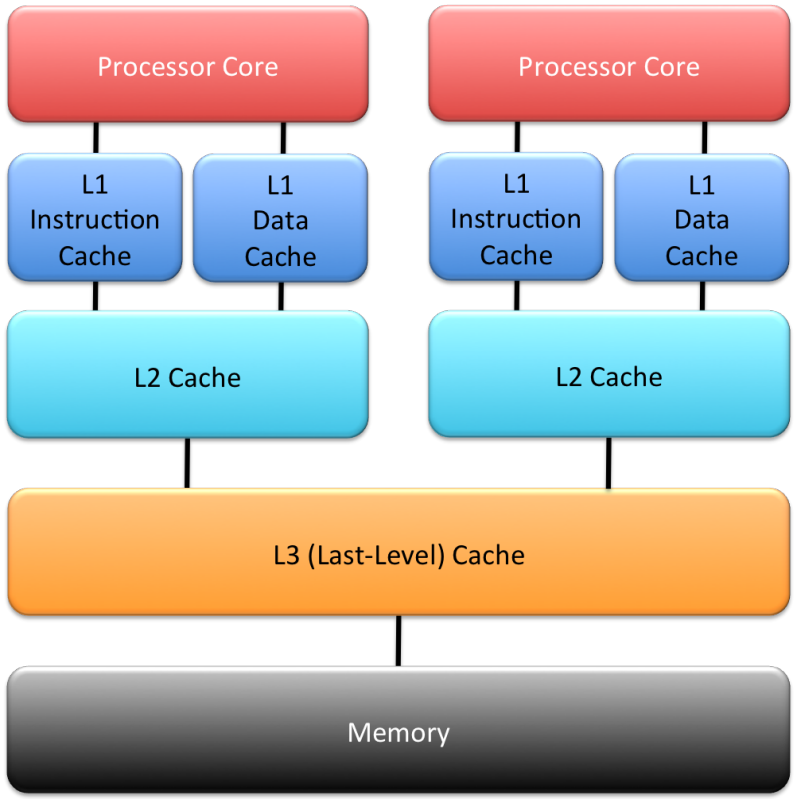


(les machines les plus puissantes actuellement contiennent plusieurs centaines de milliers de nœuds)

Vue interne d'un nœud (variable suivant le modéle de processeur et la génération utilisée):



Vue interne d'un cœur (variable suivant le modéle de processeur et la génération utilisée):



Situation actuelle (et encore pour plusieurs années):

vitesse de calcul (d'un processeur)

- $\approx$ *la mémoire interne (registres) du processeur*
- $>$ *les différents mémoires intermédiaires (caches) entre le processeur et la mémoire centrale du nœud (en gris sur la figure précédente)*
- $>>$ *vitesse de la mémoire centrale d'un nœud*
- $>>$ *vitesse du réseau qui connecte les nœuds*

Fonctionnement :

1. Il faut faire voyager les données le plus pres possible du processeur qui va les utiliser.
2. Si une donnée est utilisée n fois par le même processeur, les $n-1$ derniéres utilisations seront plus rapides.
3. Si un processeur a modifié une donnée dans une de ses mémoires, il faut répercuter cette modification dans les différentes copies de cette donnée

Cette gestion utilise du temps d'exécution (pour assurer la cohérence des différentes parties de la mémoire et leurs mises á jour correcte.)

Pour obtenir un code efficace (en temps d'exécution), il faut:

- utiliser les algorithmes les plus efficaces possible (pas couvert par ce cours)
- organiser le placement des données (améliorer la localité spatiale)
- organiser la séquence d'instruction (améliorer la localité temporelle)
- écrire les instructions pour qu'elles soient plus rapides (utilisation du // interne des processeurs)

Optimisation de la programmation séquentielle (2 séances)

Points abordés

- Modèle d'architecture matérielle
- Localités spatiale et temporelle (optimisation de l'utilisation de la mémoire cache)
- Parallélisme à l'intérieur d'un cœur
- Exemples

Localité spatiale

Règle: autant que possible, utiliser des zones mémoires proches les unes des autres dans une séquence d'instructions

But: réduire la fréquence de transferts mémoire centrale - mémoire cache

Localité temporelle

Règle: autant que possible, pour une zone mémoire, les instructions qui l'utilisent doivent s'exécuter de façon rapprochée

But: réduire la fréquence de transferts mémoire centrale - mémoire cache

Rappels de programmation parallèle: notions

Points abordés

- mémoire distribuée
- mémoire partagée
- threads
- processus

Rappels de programmation parallèle: mémoire partagée

Points abordés

- Modèle d'architecture matérielle
- Principes d'optimisation
- Cas classique : OpenMP, pthreads
- Autres

Rappels de programmation parallèle: mémoire distribuée

Points abordés

- Modèle d'architecture matérielle
- Principes d'optimisation
- Cas classique : MPI
- Autres

Programmation parallèle hybride (4 séances)

Points abordés

- Coexistence
- Modèles d'hybridation
- Cas classique : MPI - OpenMP
- Exemples
- Autres modèles (e.g. MPI+X, PGAS)

Examen (1 séance)