INFO0604
PROGRAMMATION MULTI-THREADÉE

COURS 1

PRÉSENTATION DE LA MATIÈRE, ORGANISATION ET INTRODUCTION

Pierre Delisle
Département de Mathématiques, Mécanique et Informatique
Décembre 2021

Pierre Delisle

- Maître de Conférences au département de Math-Meca-Info de l'URCA
- Mail
  - pierre.delisle@univ-reims.fr

Plan de la séance

- Description de la matière
  - Objectifs, pré-requis, organisation
- Introduction
  - Calcul parallèle ?
  - Architectures
    - Multi-coeur
  - Modèles et langages de programmation
    - Modèle de threads

Objectifs de la matière

- Objectif général
  - Acquérir des connaissances théoriques et pratiques de base en programmation multi-threadée
- Compétences spécifiques
  - Parallélisation mémoire partagée d'applications séquentielles
  - Conception d'applications parallèles
  - Programmation pThreads, Java Threads
- Pré-requis
  - Info0301 : Langage C et outils de développement associés
  - Info0403 : Systèmes d'exploitation
  - Info0401 et Info0501 sont bons à connaître aussi !

Organisation

- CM : 5 x 2 heures
  - Lundi 10h15-12h15 – Sem. 50, 1-3
  - Mardi 10h15-12h15 – Sem. 50
- TD : 5 x 2 heures
  - S6O6 : Jeudi 10h15-12h15 – Sem. 1-4
    + Jeudi 14h00-16h00 – Sem. 50
  - S6O7 : Mercredi 10h15-12h15 – Sem. 50, 1-4
- TP : 5 x 2 heures
  - À partir de la semaine 1

- Évaluation

Nature de l'évaluation	Nombre de points
Projet (Code, rapport et soutenance)	40
DST	60
Total	100

- Changements possibles sur les groupes et sur les emplois du temps
- Surveillez votre bureau virtuel et le site de la Licence Info !

Quelques informations pratiques…

- Emplois du temps et informations
  - Bureau virtuel
    - ebureau.univ-reims.fr/
  - Site Web de la Licence Informatique
    - http://www.licenceinfo.fr/
  - THOR
    - https://thor.univ-reims.fr/
  - Site Web de la scolarité
    - http://www.univ-reims.fr/ufrsciences
- Demi-groupes de TP
  - Voir sur THOR
- Supports de cours et énoncés de TD/TP
  - Moodle

Structure de la matière

- Introduction au parallélisme
  - Architectures, Algorithmes, Programmation
- Algorithmique parallèle
  - Parallélisation, conception
- Programmation multi-threadée
  - Processus et threads
  - Programmation asynchrone
  - Pthreads (mutex, variables de condition, ...)
  - Modèles de programmation multi-thread

INTRODUCTION AU CALCUL PARALLÈLE

Exercice 1

- Comptables…

Les limites du mono-processeur

- L'augmentation de la vitesse d'un processeur devient de plus en plus coûteuse et les gains sont de plus en plus limités
- Pourtant, les besoins ne diminuent pas !
- Une solution
  - dupliquer les unités de calcul
- Difficultés
  - Dégager la concurrence des codes d'application
  - Gérer les communications/accès mémoire
- Calcul parallèle
  - Exécution d'un algorithme en utilisant plusieurs processeurs plutôt qu'un seul
  - Division d'un algorithme en tâches pouvant être exécutées en même temps sur des processeurs différents
  - Le but : réduire le temps de résolution d'un problème un utilisant un ordinateur parallèle
  - 3 niveaux d'abstraction
    - Architectures, Algorithmes, Programmation

Calcul parallèle : architectures, algorithmes, programmation

- Architecture parallèle
  - ordinateur(s) comportant plusieurs processeurs et supportant la programmation parallèle
  - Plusieurs types d'architectures
    - Distribuées
    - Centralisées
- Algorithmique parallèle
  - Approches de résolution de problèmes dans un contexte d'exécution parallèle
  - Modèles algorithmiques : contextes d'exécution parallèle simplifiés pour faciliter la conception
    - PRAM
  - Analyse théorique de la performance
- Programmation parallèle
  - Programmation dans un langage permettant d'exprimer le parallélisme dans une application réelle
  - Différents niveaux d'abstraction possibles
  - La parallélisation automatique serait la solution idéale, mais c'est très difficile
  - La façon de programmer n'est pas indépendante de la machine utilisée

Le parallélisme aujourd'hui

- Le parallélisme devient de plus en plus accessible
- Les environnements de développement deviennent de plus en plus conviviaux
- Cependant...
  - Il reste du travail à faire
  - Besoin d'intelligence !
  - Besoin de solutions logicielles performantes et portables
- Situation idéale : transparence
  - Le système dégage le parallélisme à partir d'un programme séquentiel
  - Idéal, mais très difficile
- Situation à éviter : multiplicité inutile
  - Coûts de conception/développement importants
- Compromis nécessaire entre la facilité de conception/développement et la performance
- Favoriser la portabilité des applications

Architectures parallèles

Multi-ordinateur

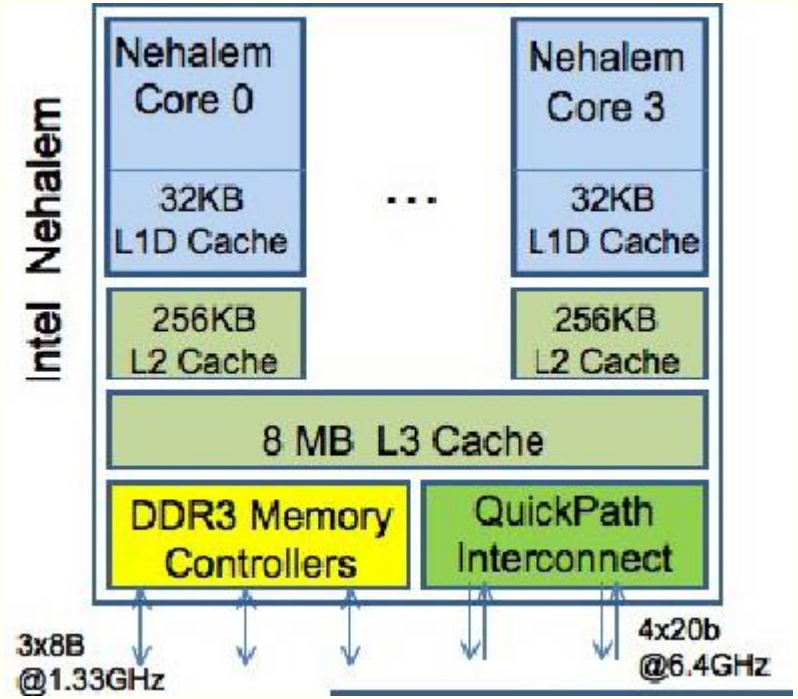
- La mémoire est physiquement distribuée entre les processeurs
- Logiquement, espaces d'adressage disjoints
- Chaque processeur n'a accès qu'à sa mémoire locale
- Interactions entre processeurs effectuées par passage de messages
- Clusters, réseaux de stations, ...

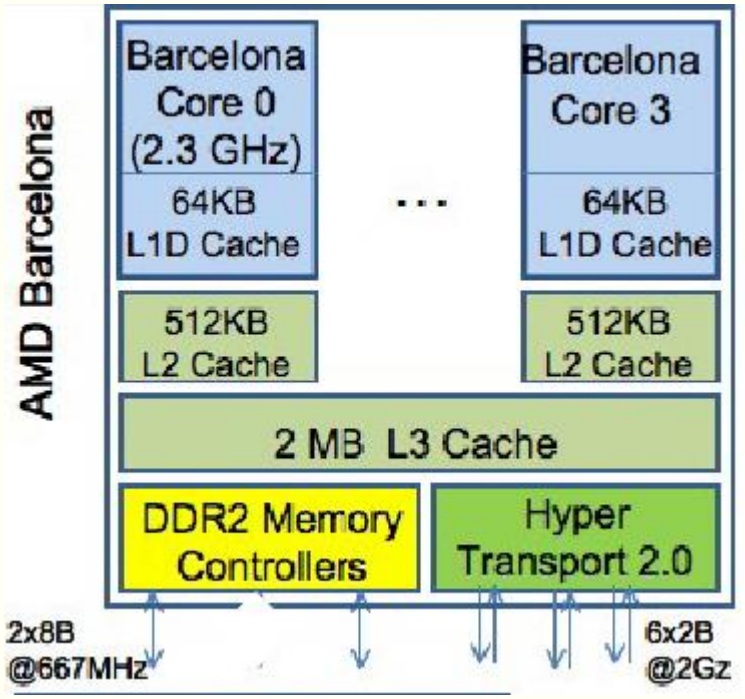
Multiprocesseur

- Ordinateur comprenant plusieurs processeurs
- Mémoire partagée
- Les caches réduisent la charge sur le bus et/ou la mémoire
- Les processeurs communiquent par des lectures/écritures sur des variables en mémoire partagée (physiquement ou virtuellement)
- SMP, Multi-coeur, ...

Architecture multi-coeur

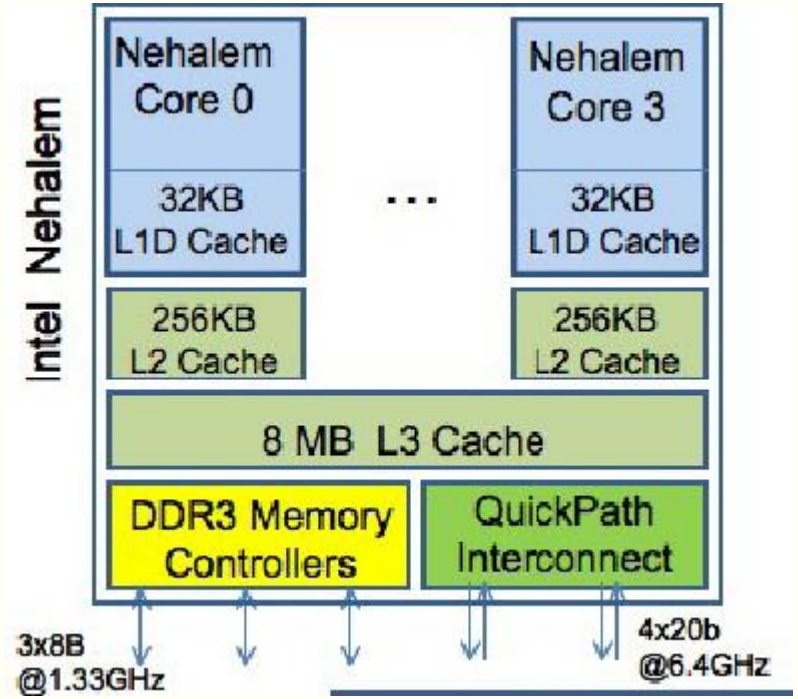
- Processeur
  - au moins 2 unités de calcul appelés cœurs
  - Augmentation de la puissance de calcul sans augmentation de la fréquence d'horloge
  - Réduction de la dissipation thermique
- Augmentation de la densité
  - Comme les coeurs sont sur le même support, la connectique reliant à la carte mère ne change pas



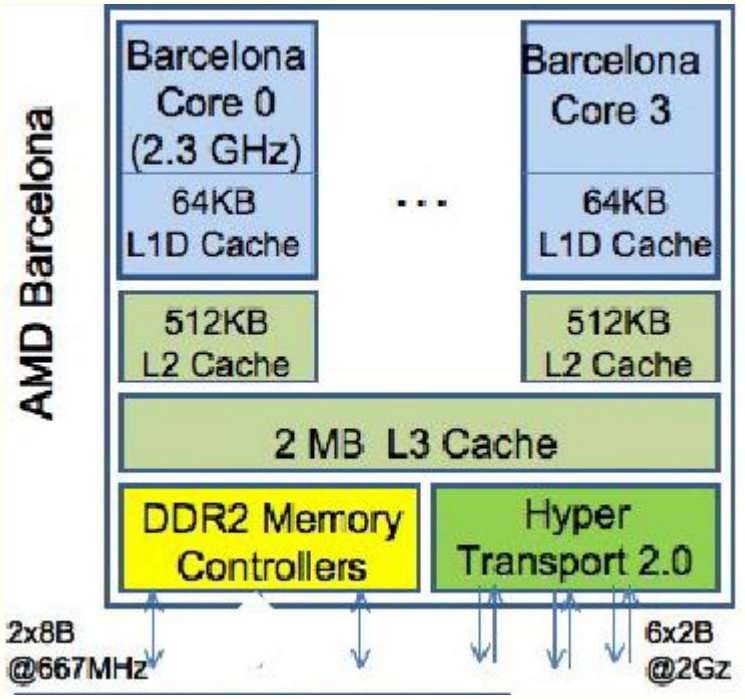


Architecture multi-coeur

- Pas de partage de caches
  - Pas de contention entre les unités de calcul
  - Communication et migration plus coûteuses
  - Passage systématique par la mémoire centrale



- Partage du cache
  - Communication plus rapide entre unités de calcul
  - Migration plus facile → on ne passe pas par la mémoire centrale
  - Problème de contention au niveau des caches
  - Problème de cohérence de cache !

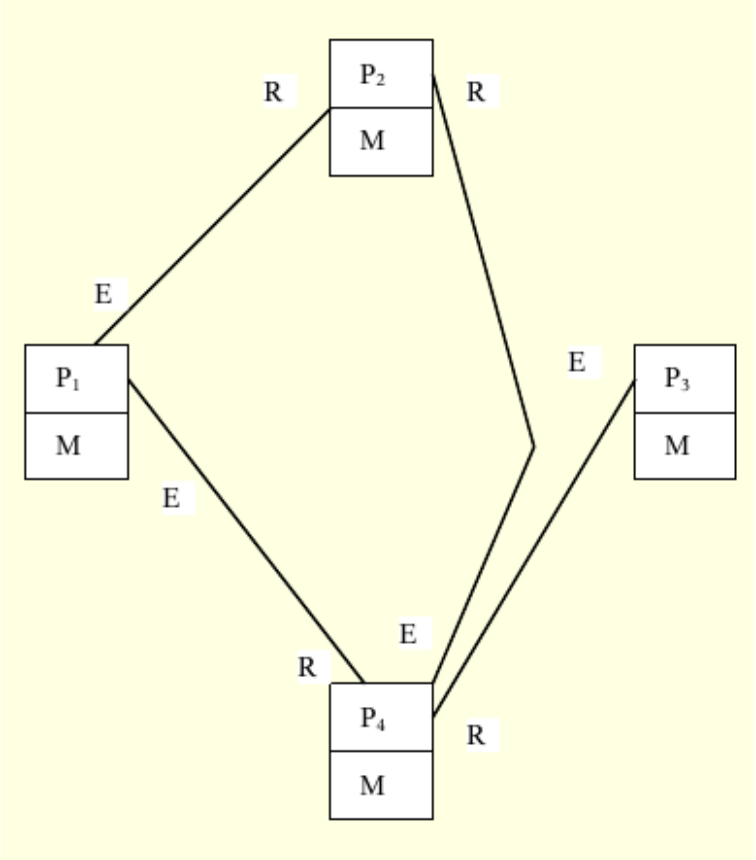


PROGRAMMATION PARALLÈLE

Mémoire distribuée, mémoire partagée

Programmation mémoire distribuée

- Modèle à passage de messages
  - Processeurs/coeurs attribués à des processus distincts
  - Chaque processus possède sa propre mémoire
  - Communications par envois et réception de messages
- Languages
  - PVM, MPI



Exemple de programme C/MPI

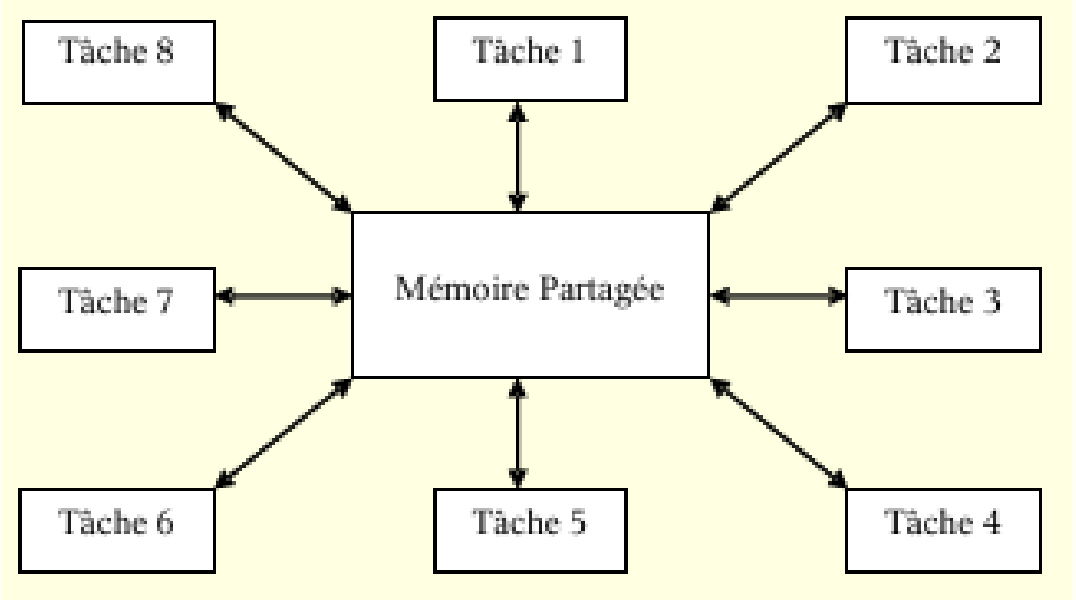
```
#include <stdio.h>
#include <mpi.h>    /*bibliotheque MPI*/

int main(int argc, char *argv[]) {
        int i;
        int id;
        int val;
        int somme;

        MPI_Init(&argc, &argv);
        MPI_Comm_rank(MPI_COMM_WORLD, &id);
        MPI_Comm_size(MPI_COMM_WORLD, &p);
        val = id + 1;
        MPI_Reduce(&val, &somme, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
        if (id == 0) {
                printf("La reduction des valeurs donne %d\n", somme);
        }
        MPI_Finalize();
        return 0;
}
```

Programmation mémoire partagée

- Modèle à mémoire partagée
  - Plusieurs processeurs/coeurs s'exécutent en parallèle
  - Processus attribués à des processeurs/coeurs distincts
  - Mémoire partagée (physiquement ou virtuellement)
  - Communications par lectures/écritures dans la mémoire partagée
- 2 principales représentations
  - Fork/join
  - Threads

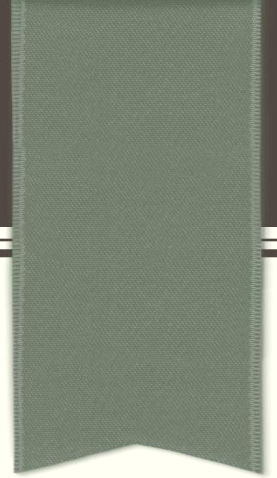


Modèle de threads

- Programme
  - ensemble de processus légers (threads) qui coopèrent afin de réaliser un objectif commun
- Thread
  - entité logique, fonction, correspondant à une partie de programme
  - pouvant s'exécuter indépendamment des autres parties
  - pouvant s'exécuter de façon concurrente
  - possédant des données locales et globales
- Un programme écrit sous forme de threads peut être exécuté séquentiellement
  - Pseudo-parallélisme
  - Partage du temps CPU entre les threads durant l'exécution
  - Changements de contexte (context switchs)
- Ce même programme peut aussi être exécuté de façon réellement parallèle
  - Répartition des threads sur les processeurs
  - Un ou plusieurs threads par processeur

Modèle de threads

- Communication par les données globales stockées dans la mémoire qui est partagée
- Fonctionnalités pour
  - La gestion des threads
    - création, destruction, etc.
  - L'ordonnancement et l'état des threads
    - actif, exécutable, en exécution, en veille, etc.
  - La synchronisation
    - sémaphores, barrières, etc.
- Principaux modèles de programmation
  - Pthreads
    - largement utilisé dans les bibliothèques des systèmes de type UNIX
  - Java Threads
    - extension du langage Java qui encapsule une classe Thread



PROCHAIN COURS

PTHREADS