

Conseil **Intranet / Extranet** Graphisme  
Audit Développement NTIC Web 2.0  
Intégration **Formation** Linux

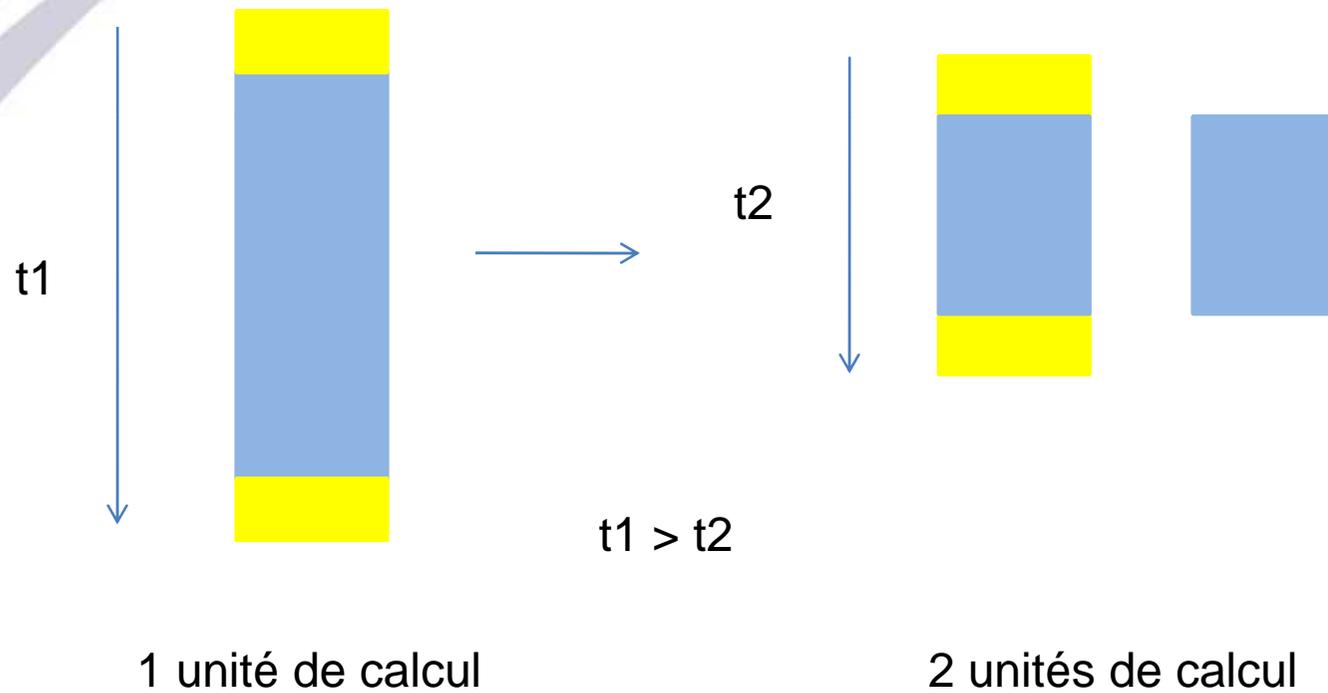


# Présentation CaSciModOT

## Performances et Architectures

- ❑ Code parallèle : Un peu de théorie
- ❑ Architectures variables :  $C(n,p)$  ?
- ❑ Quel code ? Quelle architecture ?

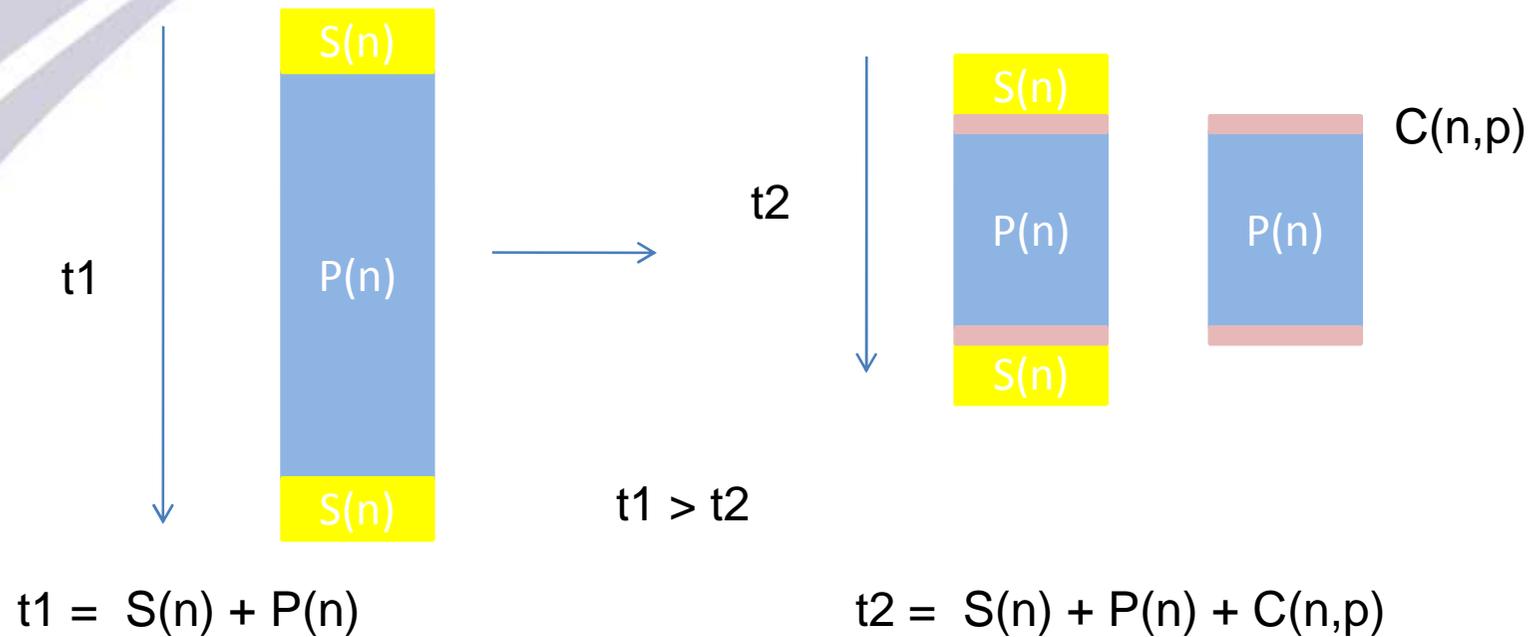
## Structure d'un code : partie parallèle / séquentielle



## Structure d'un code parallèle

1. Début
2. Lecture données, découpage des données
3. Communication des travaux aux unité de calculs
4. Calculs
5. Récupération des résultats
6. Traitements et Retour.
7. Fin

## Structure d'un code : Accélération



$$A(n, p) = \frac{S(n) + P(n)}{S(n) + P(n) / p + C(n, p)}$$

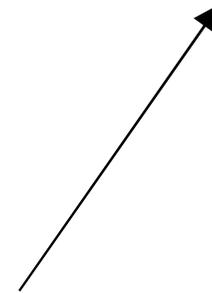
Vite, des cpus, des cpus...

$$A(n, p) = \frac{S(n) + P(n)}{S(n) + P(n) / p + C(n, p)}$$

Réduit le temps  
consacré à la  
partie parallèle



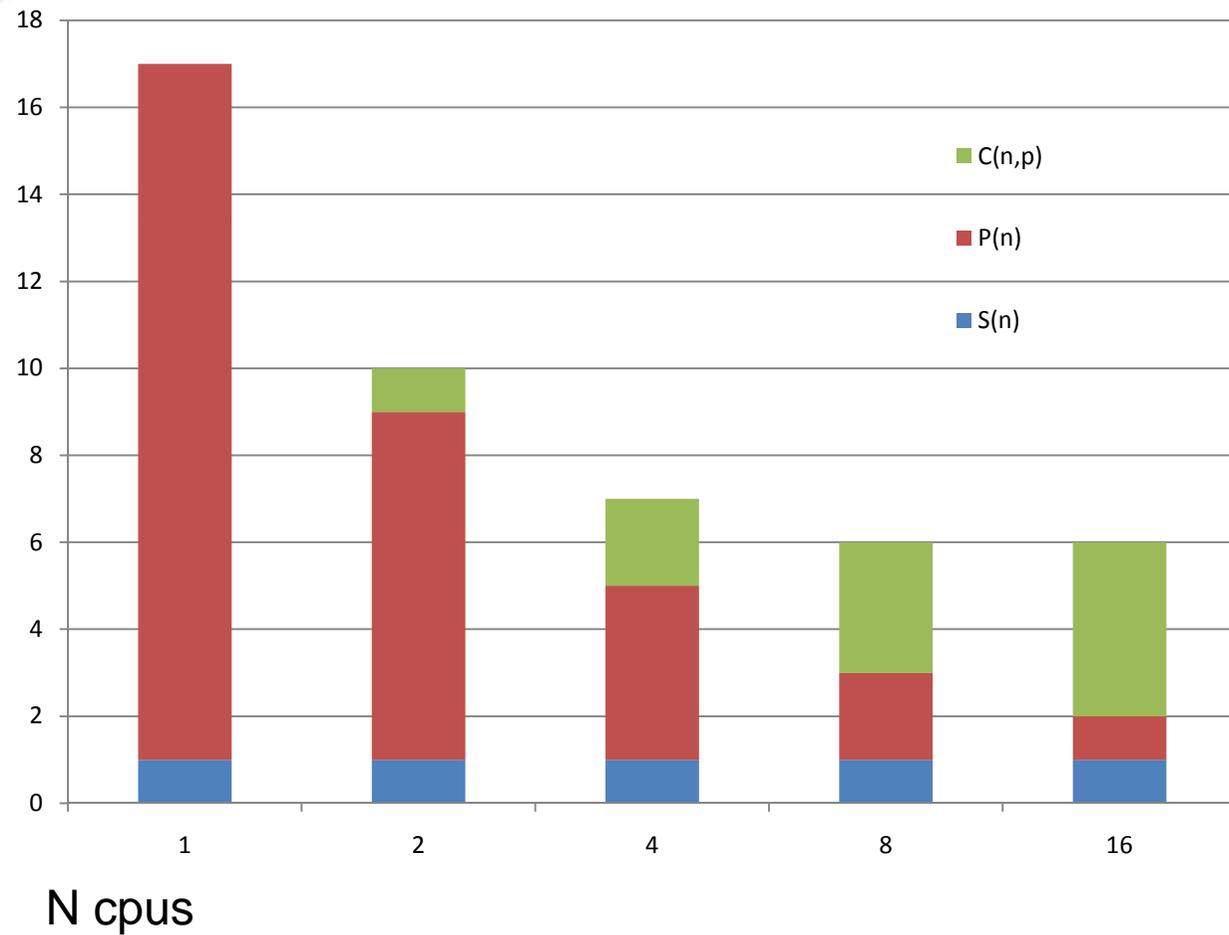
Augmente les  
coûts de  
parallélisme  
(communications)



**Augmentation du nombre  
de processeurs (p)**

## Des limites : Scalabilité...

Temps  
d'exécution

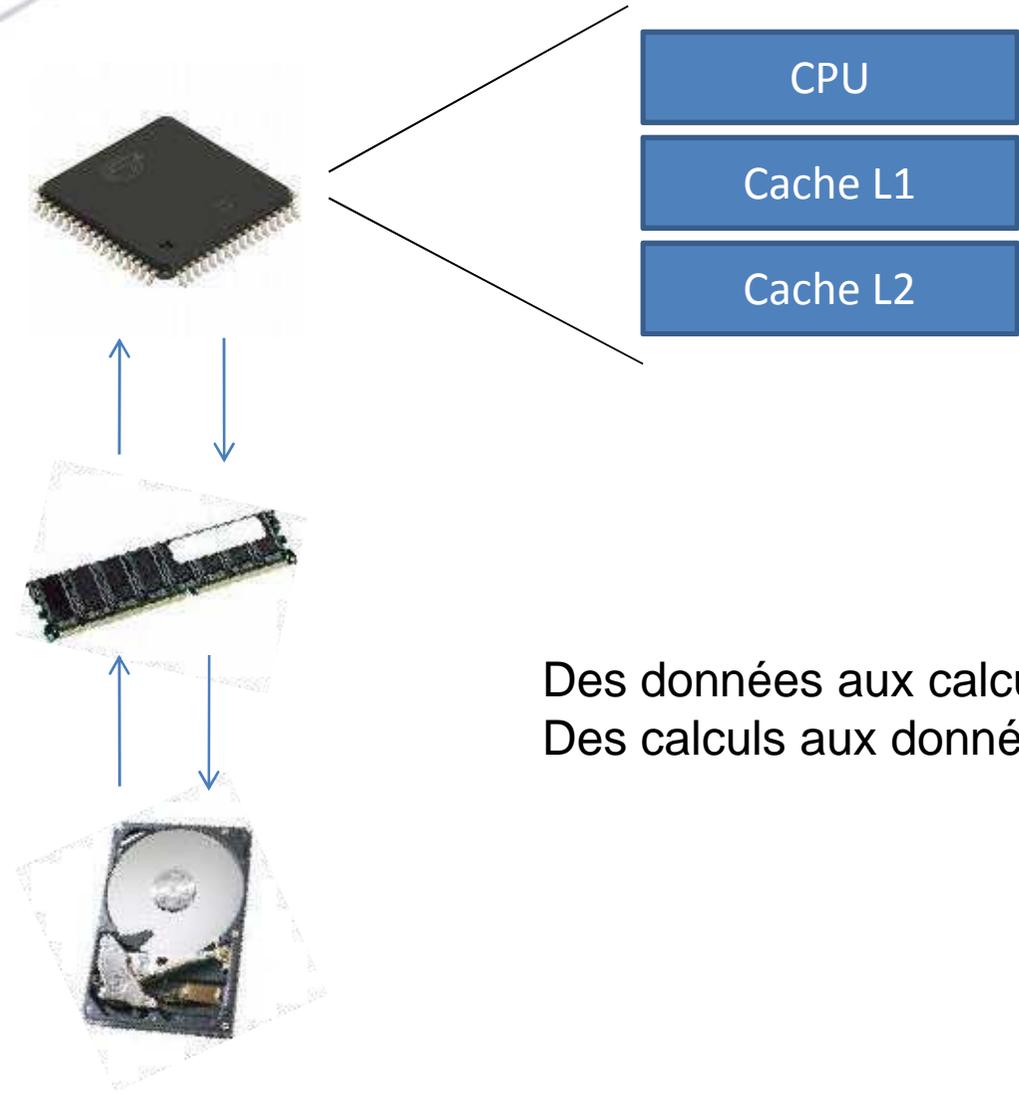


## Objectifs :

- Identifier / Comprendre  $C(n,p)$
- Réduire  $C(n,p)$
- Choisir son architecture afin de limiter le coût  $C(n,p)$

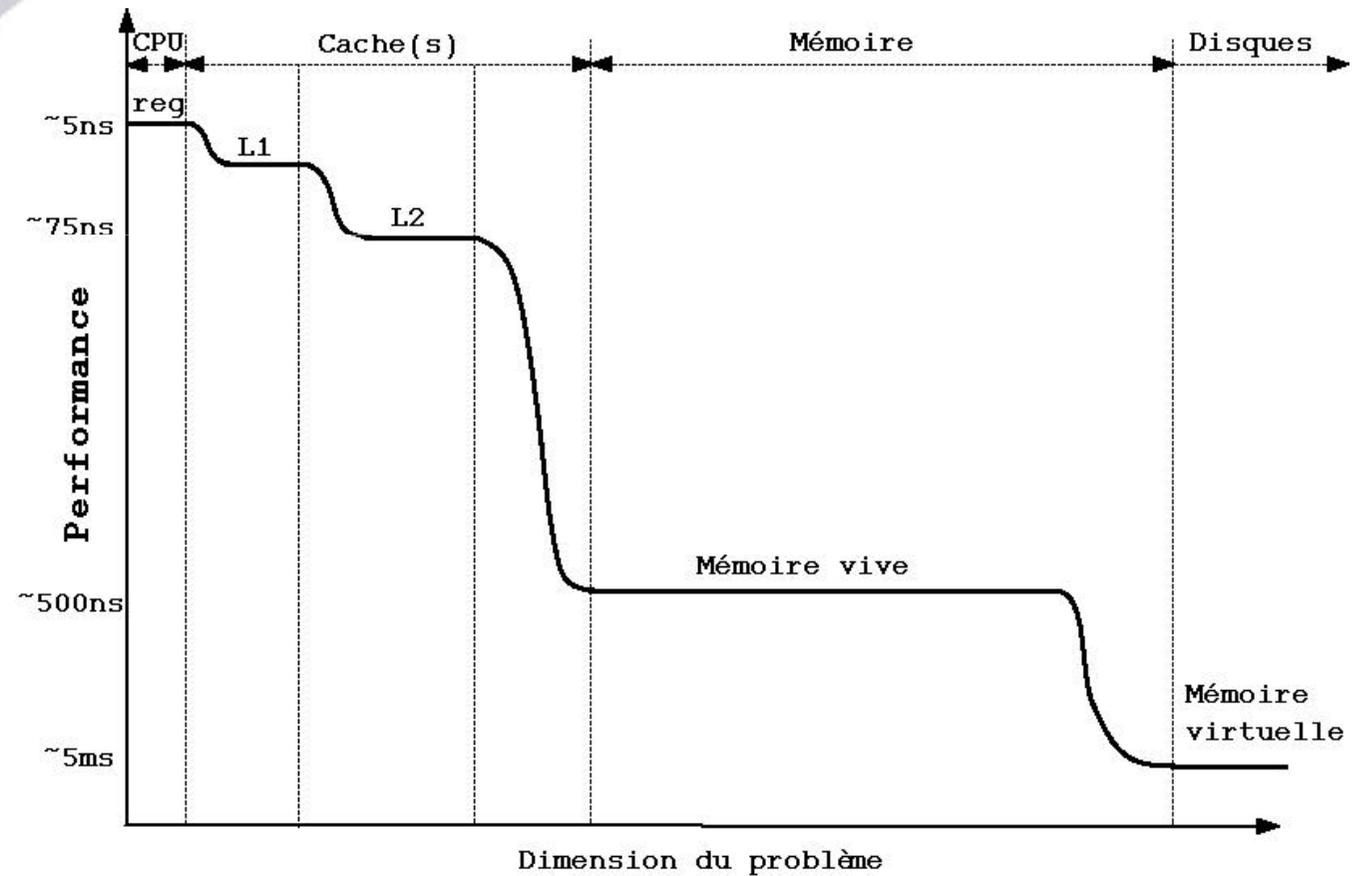
## Cherchons $C(n,p)$ ?

- Coût des bibliothèques parallèles : MPI / OpenMP ?
- Coût d'accès aux données / communications inter-nœuds ?
- Coût des accès concurrents mémoire ?
- Coût de traitement des résultats, coût IO ?

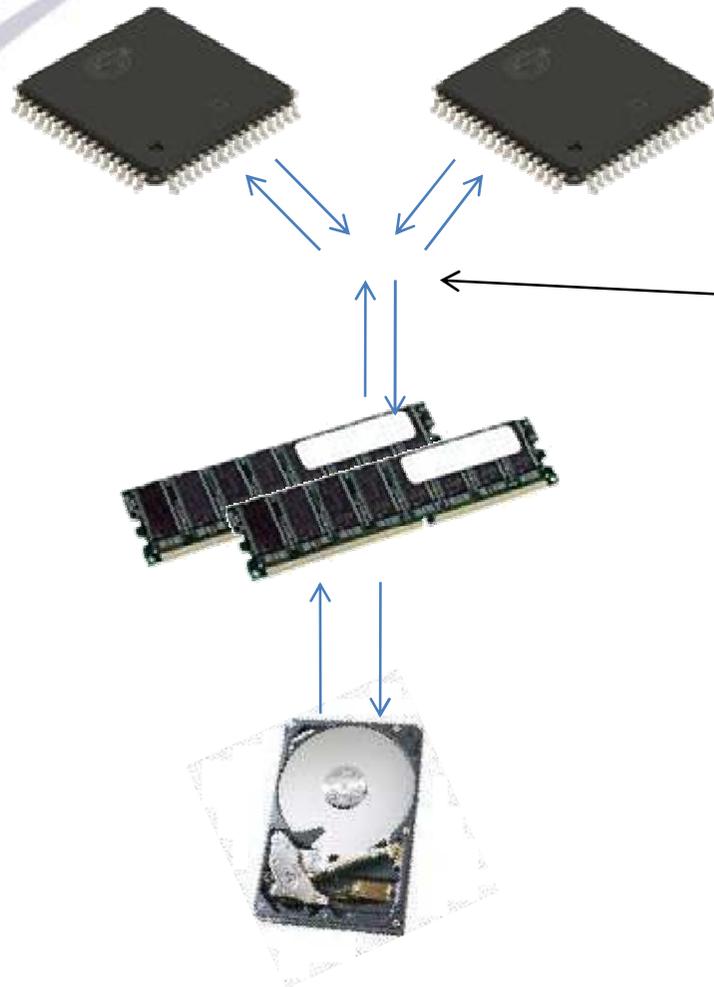


Des données aux calculs  
Des calculs aux données...

## Temps d'accès variables....



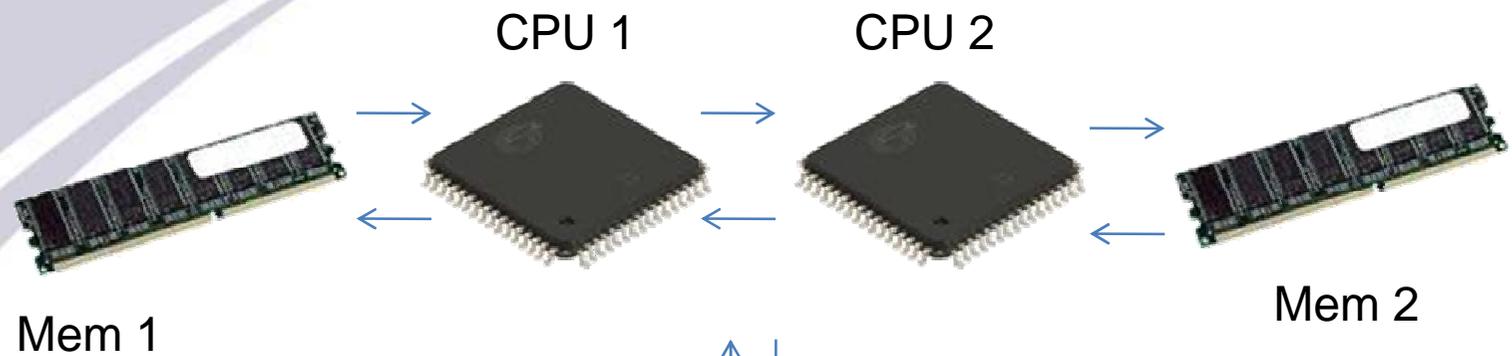
2 cpu, 2 x plus....



Goulot d'étranglement.  
Concurrence d'accès  
mémoire  
 $C(n,p)$  croit avec  $n$  et  $p$ .

Ex: ex-Architecture Intel  
Bensley - WoodCrest

## Architecture NUMA



Ex:  
HyperTransport AMD  
Barcelona  
Ou  
QuickPath Interconnect Intel  
nehalem

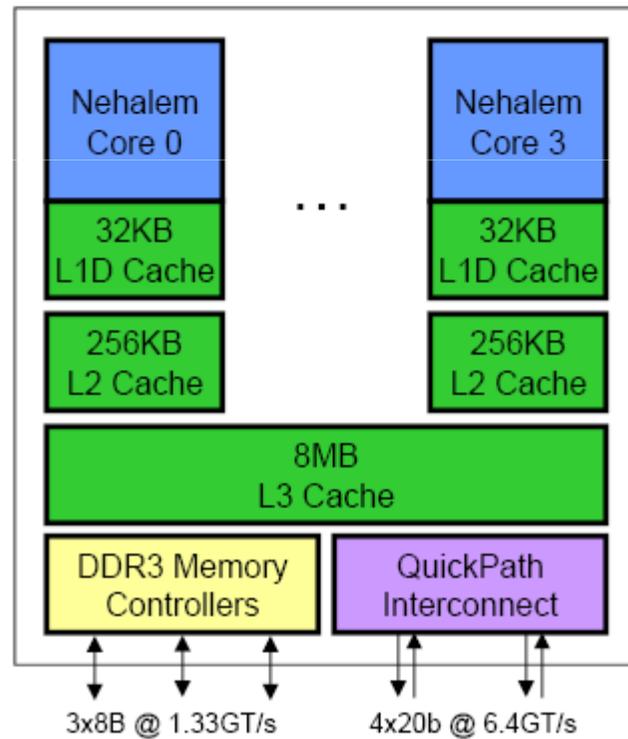
Architecture NUMA

Accès :

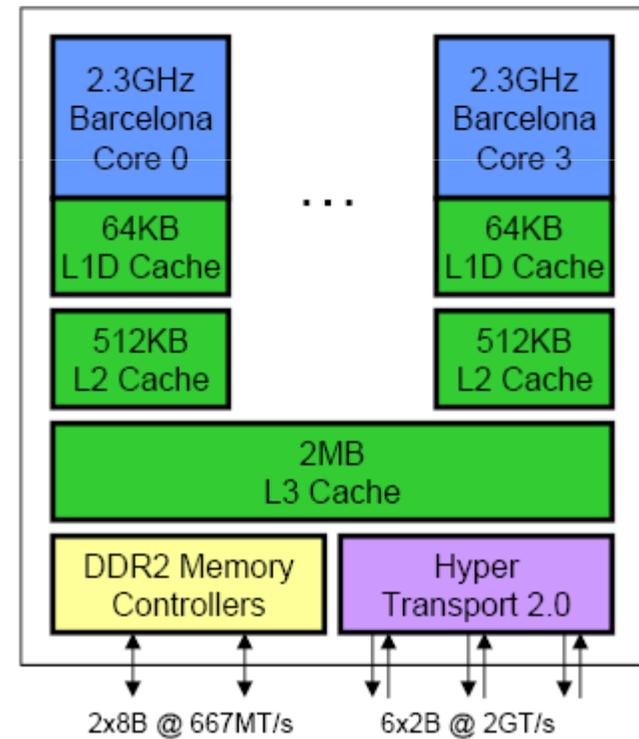
CPU1->MEM1 > CPU1->MEM2

# Architecture Multi-Core

Nehalem



Barcelona

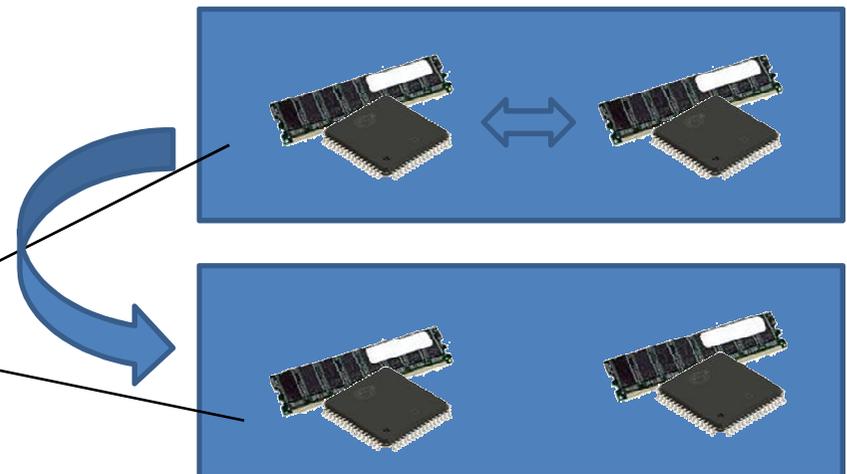


CrossBar  
C(n,p) croit  
Avec p et n.

Communications inter-serveur dans un cluster  
(communications inter-nœud)  
Utilisation Connectique Haut Débit



Latence en jeu



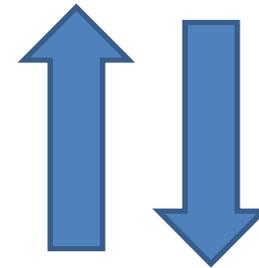
Plus je communique, plus j'ai de latences....

- ❑ Latence mémoire vive ~ 0.5  $\mu$ s
- ❑ Latence inter-nœud Infiniband/Myrinet ~ 5  $\mu$ s
- ❑ Latence inter-nœud ethernet ~ 50  $\mu$ s

$C(n,p)$  croit avec  $n$  et  $p$ .

Calculs : Génération données,  
Exploitation des résultats,  
code générateur IO

$C(n,p)$  croit avec  $n$ .



## Connaître ses besoins :

Partie  
Séquentielle  
Embarrassante  
= Peu de  
communication,  
Beaucoup  
d'actions  
Parallélisés  
SPMD :

- Calcul  
génétiques
- Traitement  
d'images
- Calculs Monte  
Carlo

Problèmes  
accès sur le  
calcul :

Besoins  
d'unités de  
calcul pour  
faire tourner un  
modèle et  
d'augmenter  
la granularité,  
précision :  
Code  
Générateur IO  
: CFD, RDM,  
Crash Test,,  
Astrophysique

Problèmes  
accès sur la  
donnée :

Besoin de  
traiter  
beaucoup de  
données, Code  
IO  
consommateur  
:  
- Climat,  
physique  
nucléaire,  
cartographie  
céleste...

## Réduire $C(n, p)$ ?

- Adapter  $n$  en fonction de  $p$  (de l'architecture) : limiter les communications (dimensionnement des échanges mémoires).
- Jouer avec les affinités mémoires / CPU : connaître les surcoûts multi-core
- Trouver une solution pour la gestion IO ?

## Comment procéder ?

- Benchmark (faire évoluer n et p), sonder son code
- Chercher les éléments limitants : accès concurrents mémoires, communications : latences mémoire/réseau, attentes IO ?
- Analyser le comportement du code parallèle mis en place (Maître Esclave, Pipelining, Equilibrage de charge...), optimiser les méthodes de parallélisation.

## Quelle architecture choisir ?

- Travailler en mémoire partagée (SMP), distribuée (cluster) ?
- Si cluster ? Quelle connectique ai-je besoin ?
- Calcul EP : recours à des processeurs « spécialisés », GPU, FPGA ?
- IO générateur/consommateur : Trouver une solution : (solution matérielle (raid, baie), solutions logiciels : Bibliothèques parallèles IO, FS distribués...).

## Conclusion : Quelle attitude adopter ?

- Connaître son architecture : Benchmark.
- Connaître les comportements de son code : Benchmark.
- Dimensionner son code (n, p) pour avoir une efficacité satisfaisante : frustration du chercheur.
- Connaître son architecture, dompter l'exécution de son code : Utilisation affinité mémoire / CPU – nœuds IO

Merci

Questions ?



Groupe Eskape  
Situé à Tours

[www.connectikup.com/hpc](http://www.connectikup.com/hpc)



Pascal Pucci  
[ppucci@eskape.fr](mailto:ppucci@eskape.fr)  
Division HPC

