Mais pourquoi donc est-ce que mon code ne va pas si vite que ça? Intensité arithmétique, Roofline model, Numa etc...

Roland Denis (Thierry Dumont)

Institut Camille Jordan

8 Février 2018

Le parallélisme en mémoire partagée expliqué en 2 minutes.

Machines actuelles :

- $1, 2, 4...$ processeurs,
- chaque processeur a n cœurs, avec $n = 2, 4, 8, 10, 12, \ldots$.

Bientôt : une centaine de cœurs par machine.

Le parallélisme en mémoire partagée expliqué en 2 minutes.

Machines actuelles :

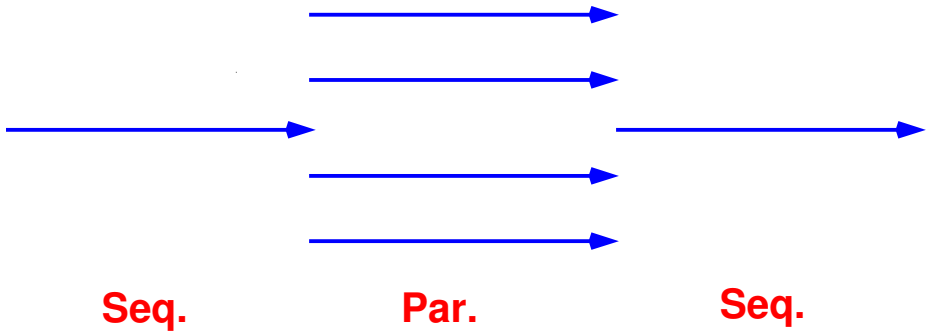
- $1, 2, 4...$ processeurs,
- chaque processeur a n cœurs, avec $n = 2, 4, 8, 10, 12, \ldots$.

Bientôt : une centaine de cœurs par machine.

Parallélisme obligatoire !

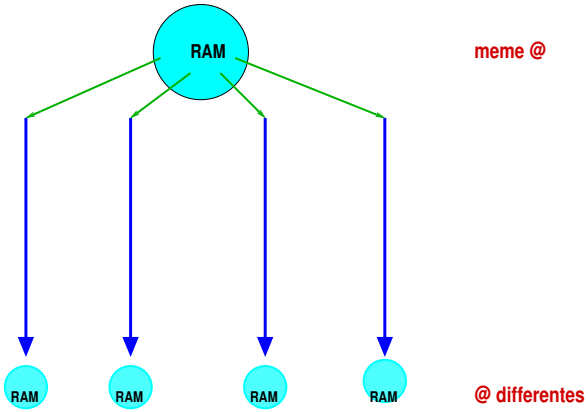
Le parallélisme en mémoire partagée expliqué en 2 minutes.

Processus légers (threads) :



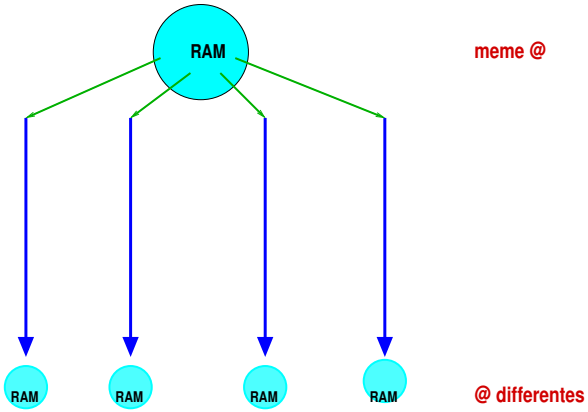
Le parallélisme en mémoire partagée expliqué en 2 minutes.

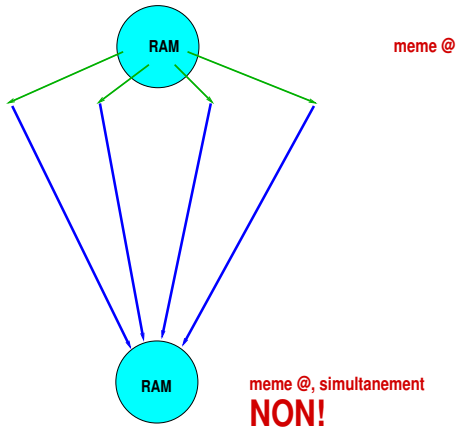
Partage de la mémoire (sans sécurité).



Le parallélisme en mémoire partagée expliqué en 2 minutes.

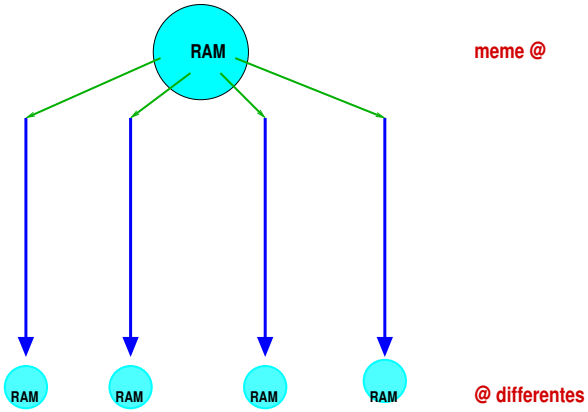
Partage de la mémoire (sans sécurité).

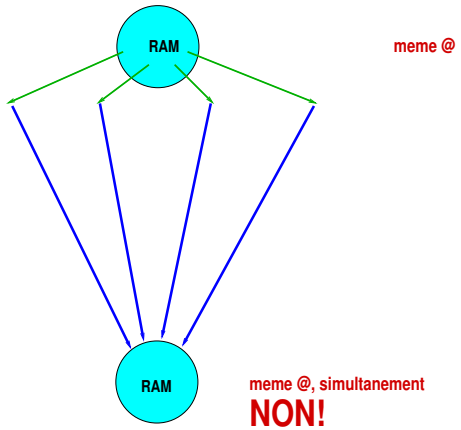




Le parallélisme en mémoire partagée expliqué en 2 minutes.

Partage de la mémoire (sans sécurité).

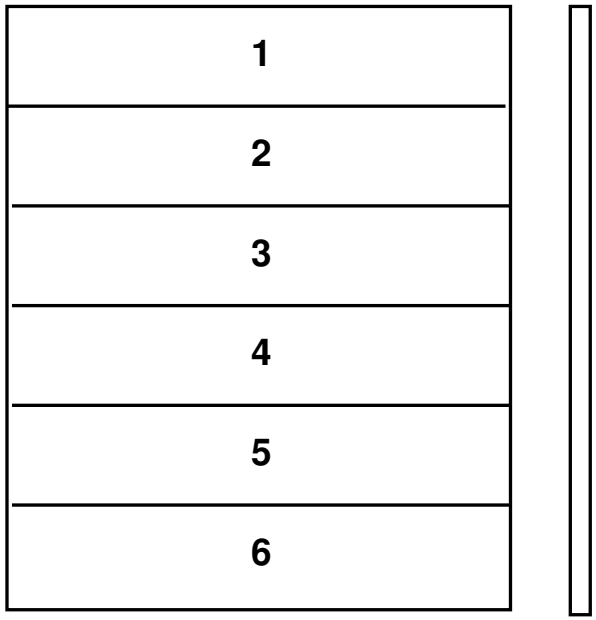




Standard : Open MP, mais aussi TBB (C++).

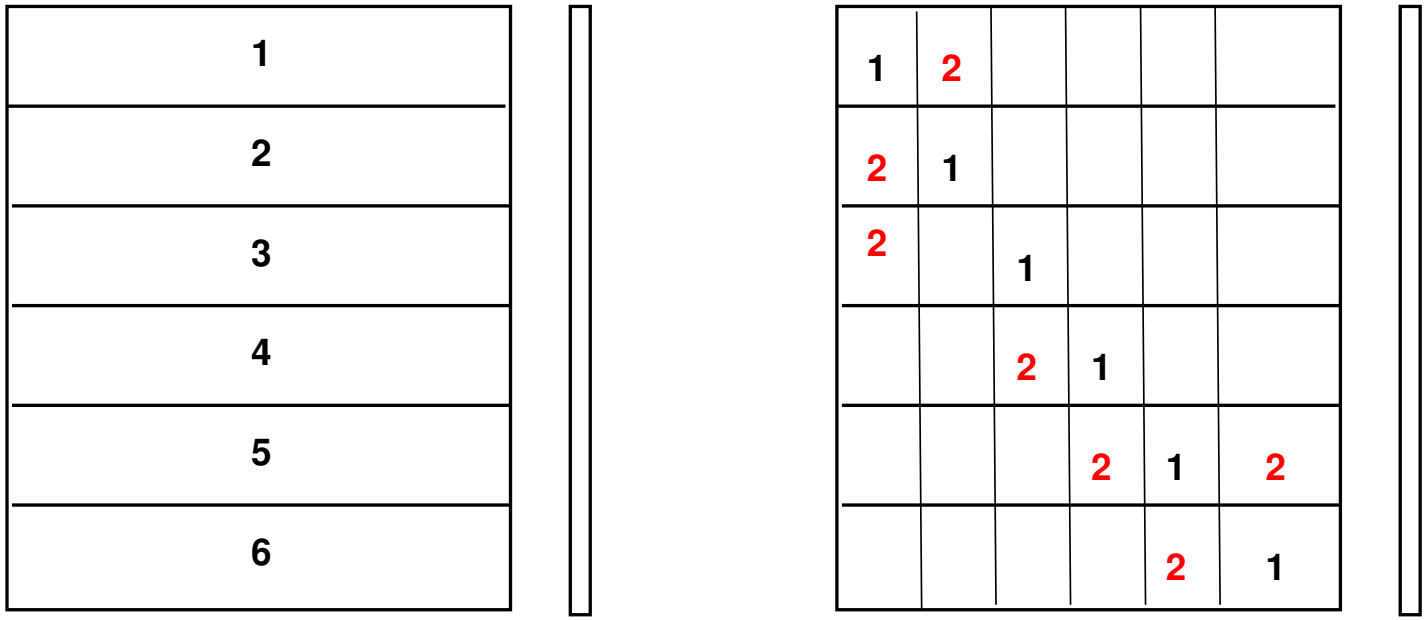
Le parallélisme en mémoire partagée expliqué en 2 minutes.

Exemple : le produit matrice x vecteur.



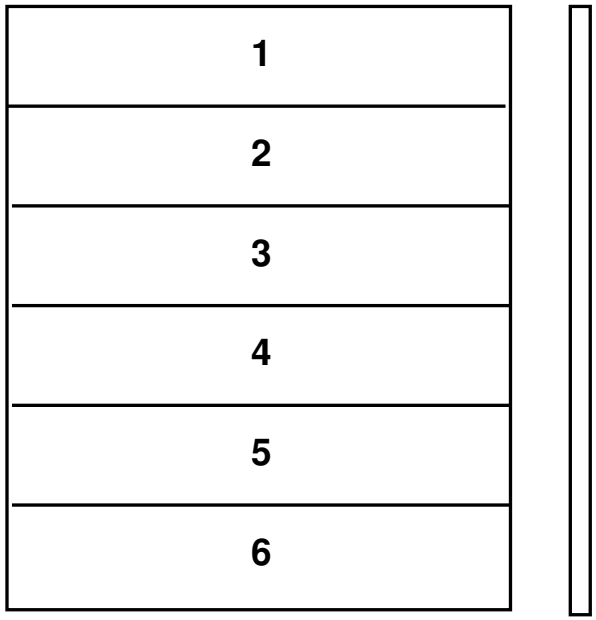
Le parallélisme en mémoire partagée expliqué en 2 minutes.

Exemple : le produit matrice x vecteur.

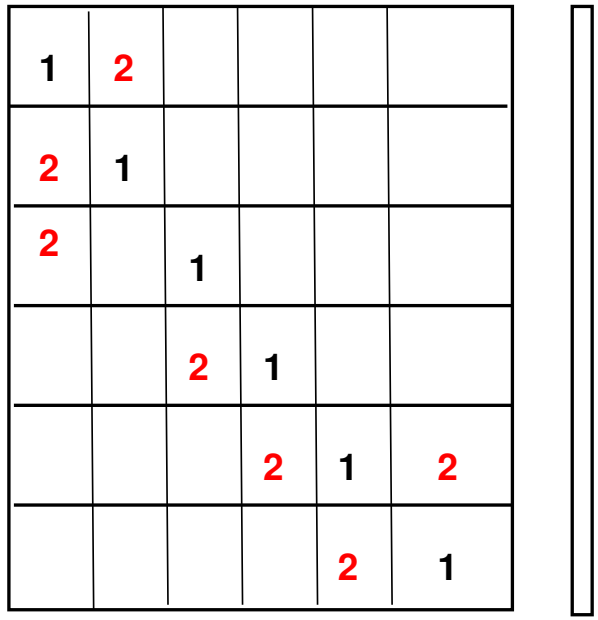


Le parallélisme en mémoire partagée expliqué en 2 minutes.

Exemple : le produit matrice x vecteur.



OK !



NON !

Performances d'une machine : rêves et réalité

On compte les flops : $\times, +-$.
Attention aux divisions (plus lentes) !

Performances d'une machine : rêves et réalité

On compte les flops : $\times, +-$.
Attention aux divisions (plus lentes) !

Exemple :
machine Intel « Sandy bridge » 2 processeurs, avec chacun 8 cœurs, tournant à 2.6 Ghz ($2.6\ 10^9$ cycles/seconde).

Performances d'une machine : rêves et réalité

On compte les flops : $\times, +-$.
Attention aux divisions (plus lentes) !

Exemple :
machine Intel « Sandy bridge » 2 processeurs, avec chacun 8 cœurs, tournant à 2.6 Ghz ($2.6\ 10^9$ cycles/seconde).

Aspect SIMD (Single Instruction, Multiple Data) :

Instructions AVX :

En un cycle d'horloge, faire :
$y_i = a_i\, x_i + b_i,\ i = 1, \ldots, 4$ (8 flops).

Performances d'une machine : rêves et réalité

On compte les flops : $\times, +-$.
Attention aux divisions (plus lentes) !

Exemple :
machine Intel « Sandy bridge » 2 processeurs, avec chacun 8 cœurs, tournant à 2.6 Ghz ($2.6\ 10^9$ cycles/seconde).

Aspect SIMD (Single Instruction, Multiple Data) :

Instructions AVX :

En un cycle d'horloge, faire :
$y_i = a_i\, x_i + b_i,\ i = 1, \ldots, 4$ (8 flops).

La performance *peak* est donc de :

2×8

Performances d'une machine : rêves et réalité

On compte les flops : $\times, +-$.
Attention aux divisions (plus lentes) !

Exemple :
machine Intel « Sandy bridge » 2 processeurs, avec chacun 8 cœurs, tournant à 2.6 Ghz ($2.6\ 10^9$ cycles/seconde).

Aspect SIMD (Single Instruction, Multiple Data) :

Instructions AVX :

En un cycle d'horloge, faire :
$y_i = a_i\, x_i + b_i,\ i = 1, \ldots, 4$ (8 flops).

La performance *peak* est donc de :

$2 \times 8 \times 2.6\ 10^9$

Performances d'une machine : rêves et réalité

On compte les flops : $\times, +-$.
Attention aux divisions (plus lentes) !

Exemple :
machine Intel « Sandy bridge » 2 processeurs, avec chacun 8 cœurs, tournant à 2.6 Ghz ($2.6\ 10^9$ cycles/seconde).

Aspect SIMD (Single Instruction, Multiple Data) :

Instructions AVX :

En un cycle d'horloge, faire :
$y_i = a_i\, x_i + b_i,\ i = 1, \ldots, 4$ (8 flops).

La performance *peak* est donc de :

$2 \times 8 \times 2.6\ 10^9 \times 8$

Performances d'une machine : rêves et réalité

On compte les flops : $\times, +-$.
Attention aux divisions (plus lentes) !

Exemple :
machine Intel « Sandy bridge » 2 processeurs, avec chacun 8 cœurs, tournant à 2.6 Ghz ($2.6\ 10^9$ cycles/seconde).

Aspect SIMD (Single Instruction, Multiple Data) :

Instructions AVX :

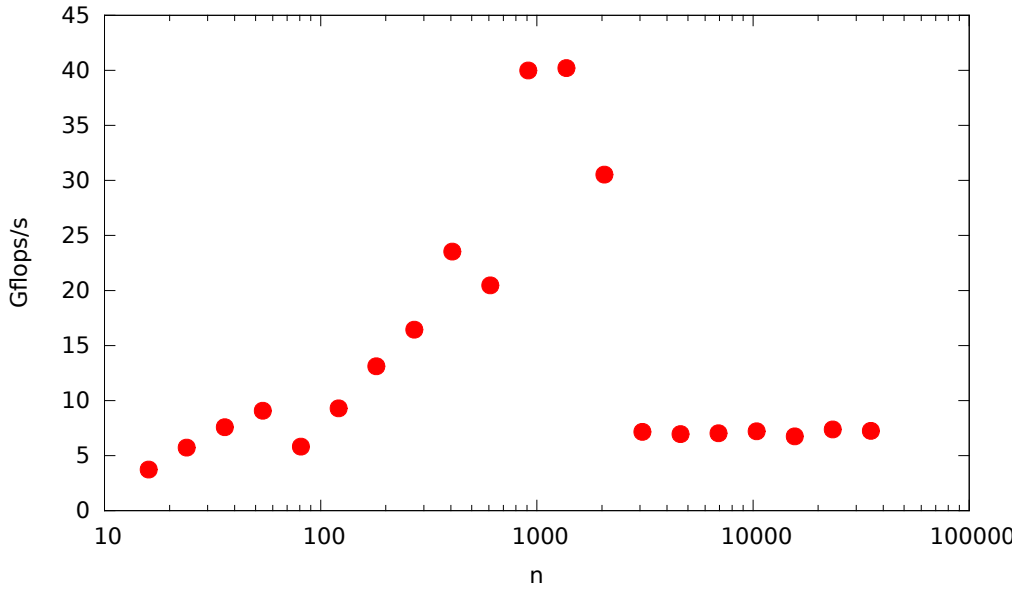
En un cycle d'horloge, faire :
$y_i = a_i\, x_i + b_i,\ i = 1, \ldots, 4$ (8 flops).

La performance *peak* est donc de :

$2 \times 8 \times 2.6\ 10^9 \times 8 = 332$ Gflops/seconde.

Essai : produit matrice x vecteur

Blas Intel, parallèle.



Nombre d'opérations :

- produit scalaire de 2 vecteurs de taille n : $2n$ flops.
- produit matrice($n \times n$) x vecteur n : n produits scalaires $=> 2n^2$ flops.

Bande passante

Un calcul ne peut être effectué que si :

1. les opérandes sont disponibles,
2. on peut écrire le résultat.

Bande passante

Un calcul ne peut être effectué que si :

1. les opérandes sont disponibles,
2. on peut écrire le résultat.

$=>$ **la bande passante entre (processeur / cache) et la mémoire limite les performances.**

Bande passante

Un calcul ne peut être effectué que si :

1. les opérandes sont disponibles,
2. on peut écrire le résultat.

$=>$ **la bande passante entre (processeur / cache) et la mémoire limite les performances.**

Intensité arithmétique

I_a = nombre d'operations / quantité de mémoire échangée.

Bande passante

Un calcul ne peut être effectué que si :

1. les opérandes sont disponibles,
2. on peut écrire le résultat.

$=>$ **la bande passante entre (processeur / cache) et la mémoire limite les performances.**

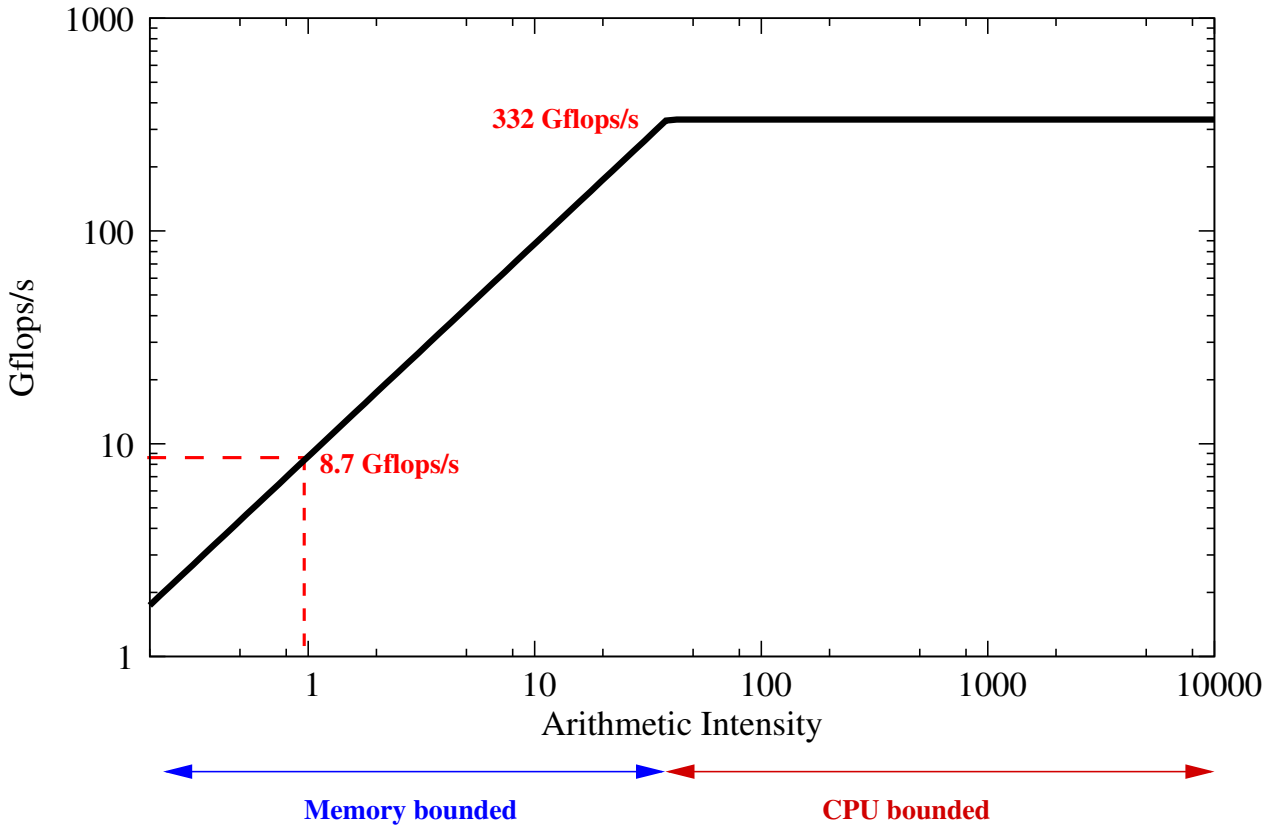
Intensité arithmétique

I_a = nombre d'operations / quantité de mémoire échangée.

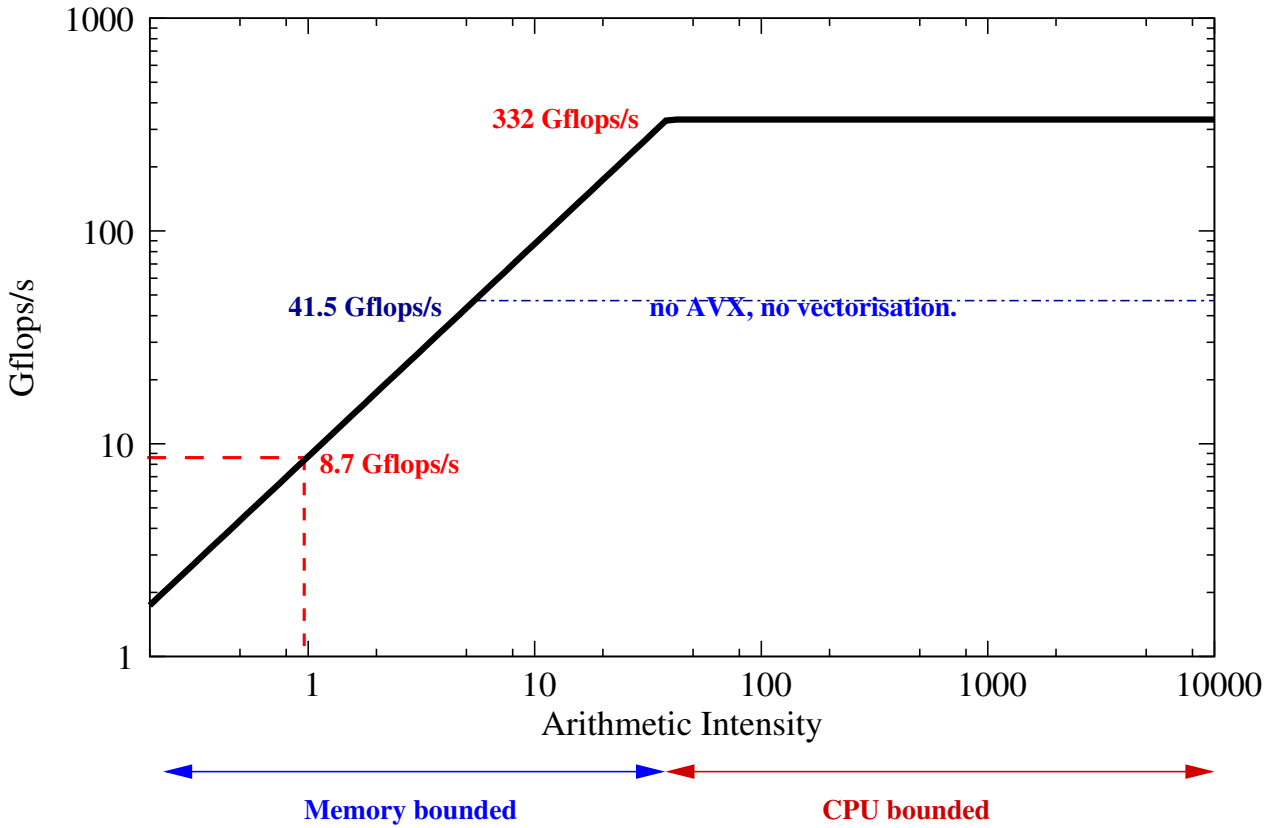
GFlops/sec atteignable =

min(Performance « peak », Bande passante $\times$ I_a).

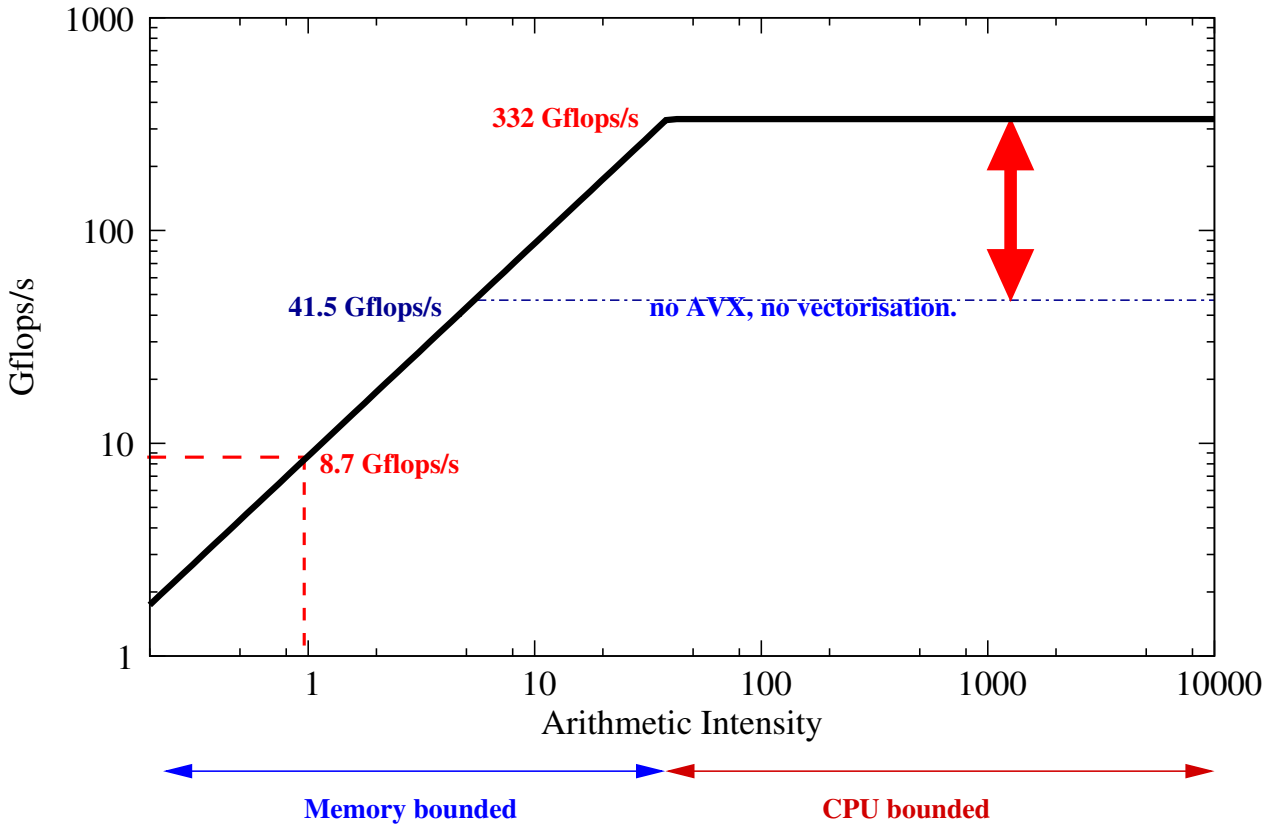
Le « Roofline Model »



Le « Roofline Model »



Le « Roofline Model »



Williams S. et al: *Roofline : An Insightful Visual Performance Model for Multicore Architectures* – Commun. ACM, 2009.

Intensité arithmétique et Roofline Model : quelques exemples

Unité utilisée : double.

1. Produit scalaire $s = \sum_{i=1}^{n} x_i.y_i$: $I_a = 1$.

Intensité arithmétique et Roofline Model : quelques exemples

Unité utilisée : double.

1. Produit scalaire $s = \sum_{i=1}^{n} x_i.y_i$: $I_a = 1$.
2. Appliquer le stencil du Laplacien à 7 en dimension 3 : $I_a = 8/8 = 1$.

Intensité arithmétique et Roofline Model : quelques exemples

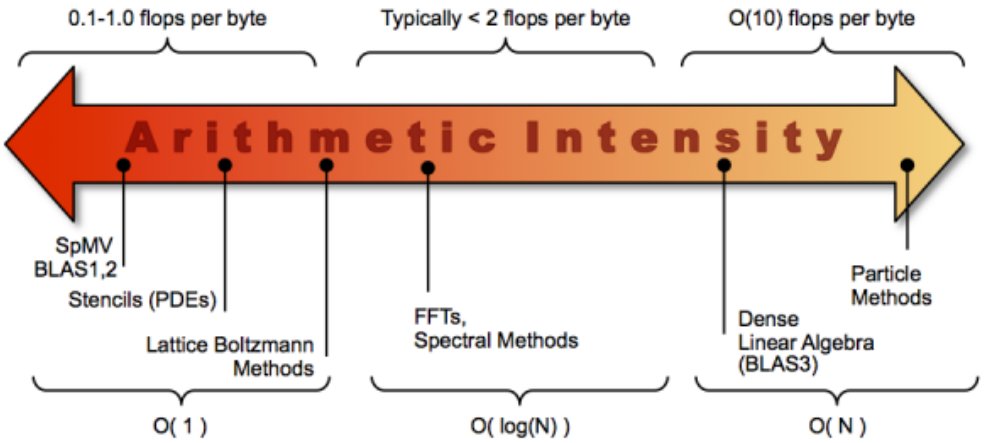
Unité utilisée : double.

1. Produit scalaire $s = \sum_{i=1}^{n} x_i.y_i$: $I_a = 1$.
2. Appliquer le stencil du Laplacien à 7 en dimension 3 : $I_a = 8/8 = 1$.
3. Produit Matrice $\times$ Matrice $C = A.B$: $I_a = 2\ n^3/3\ n^2 = \mathcal{O}(n)$.

Intensité arithmétique et Roofline Model : quelques exemples

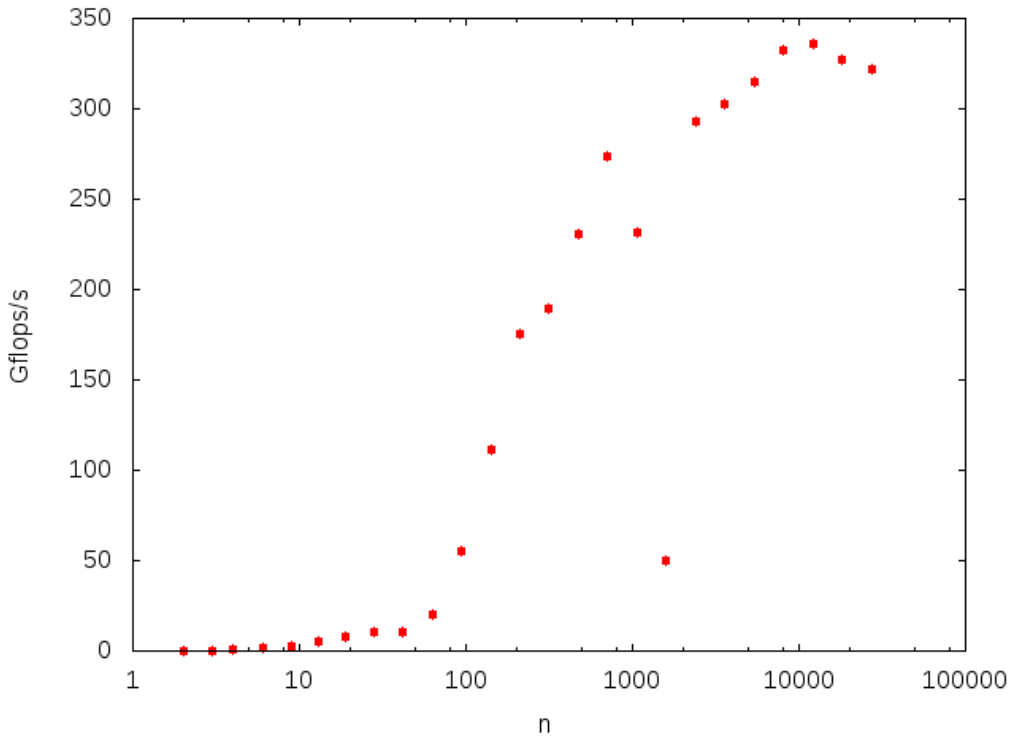
Unité utilisée : double.

1. Produit scalaire $s = \sum_{i=1}^{n} x_i.y_i$: $I_a = 1$.
2. Appliquer le stencil du Laplacien à 7 en dimension 3 : $I_a = 8/8 = 1$.
3. Produit Matrice $\times$ Matrice $C = A.B$: $I_a = 2\ n^3/3\ n^2 = \mathcal{O}(n)$.



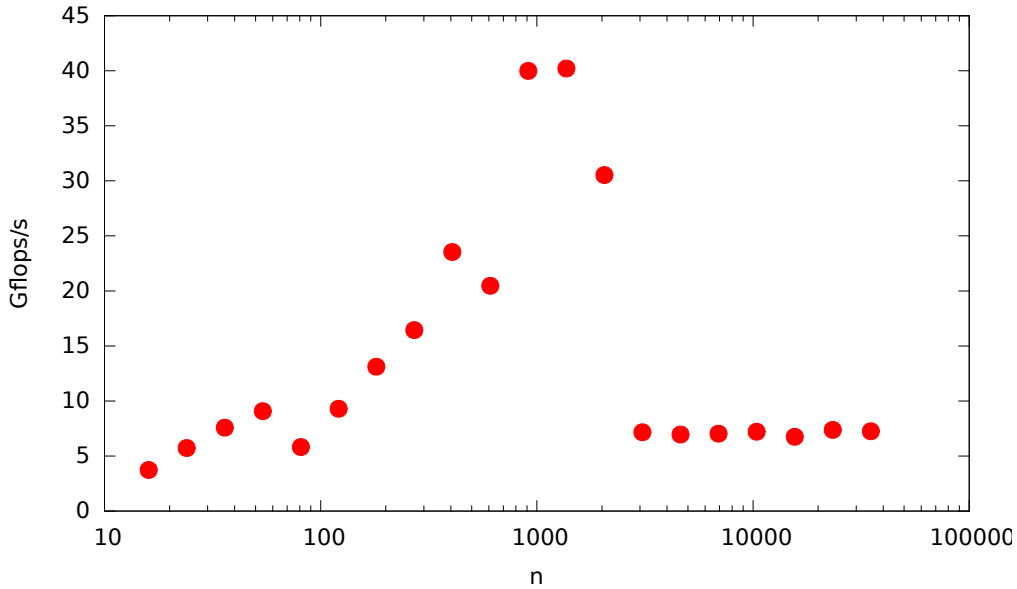
Intensité arithmétique : experiences

Matrix $\times$ Matrix product (DGEMM, Intel mkl parallel version) :



Intensité arithmétique : experiences

Produit Matrice $\times$ Vecteur (DGEMV, Intel mkl parallel version) :



Intensité arithmétique : expériences

Appiquer le stencil du laplacien en 3-d (7 points), stocké en matrice CSR :

Intensité arithmétique : expériences

Appiquer le stencil du laplacien en 3-d (7 points), stocké en matrice CSR :

	0	1	2	3	4
0	2.0		3.5		6.7
1		8.2		9.2	
2		1.1	2.8		
3	3.0		1.5	4.5	
4		2.5		8.9	

	0	1	2	3	4	5
rowptr	0	3	5	7	10	12

	0	1	2	3	4	5	6	7	8	9	10	11
colind	0	2	4	1	3	1	2	0	2	3	1	3

	0	1	2	3	4	5	6	7	8	9	10	11
values	2.0	3.5	6.7	8.2	9.2	1.1	2.8	3.0	1.5	4.5	2.5	8.9

Intensité arithmétique : expériences

Appiquer le stencil du laplacien en 3-d (7 points), stocké en matrice CSR :

	0	1	2	3	4
0	2.0		3.5		6.7
1		8.2		9.2	
2		1.1	2.8		
3	3.0		1.5	4.5	
4		2.5		8.9	

	0	1	2	3	4	5
rowptr	0	3	5	7	10	12

	0	1	2	3	4	5	6	7	8	9	10	11
colind	0	2	4	1	3	1	2	0	2	3	1	3

	0	1	2	3	4	5	6	7	8	9	10	11
values	2.0	3.5	6.7	8.2	9.2	1.1	2.8	3.0	1.5	4.5	2.5	8.9

On fait l'hypothèse (raisonnable) que :
`sizeof(double) = 2 sizeof(int)`.

- Algorithme : Mémoire $= 37/2$ doubles ; Flops $= 13 \Longrightarrow$ $I_a \simeq 0.7$.

Intensité arithmétique : expériences

Appiquer le stencil du laplacien en 3-d (7 points), stocké en matrice CSR :

	0	1	2	3	4
0	2.0		3.5		6.7
1		8.2		9.2	
2		1.1	2.8		
3	3.0		1.5	4.5	
4		2.5		8.9	

	0	1	2	3	4	5
rowptr	0	3	5	7	10	12

	0	1	2	3	4	5	6	7	8	9	10	11
colind	0	2	4	1	3	1	2	0	2	3	1	3

	0	1	2	3	4	5	6	7	8	9	10	11
values	2.0	3.5	6.7	8.2	9.2	1.1	2.8	3.0	1.5	4.5	2.5	8.9

On fait l'hypothèse (raisonnable) que :
`sizeof(double) = 2 sizeof(int)`.

- Algorithme : Mémoire $= 37/2$ doubles ; Flops $= 13 \Longrightarrow$ $I_a \simeq 0.7$.
- Machine : Mémoire : 8.73 Giga doubles/s
  $=>$ Atteignable $= 0.7 \times 8.73 = 6.11$ Gflops.

Intensité arithmétique : expériences

Appiquer le stencil du laplacien en 3-d (7 points), stocké en matrice CSR :

	0	1	2	3	4
0	2.0		3.5		6.7
1		8.2		9.2	
2		1.1	2.8		
3	3.0		1.5	4.5	
4		2.5		8.9	

	0	1	2	3	4	5
rowptr	0	3	5	7	10	12

	0	1	2	3	4	5	6	7	8	9	10	11
colind	0	2	4	1	3	1	2	0	2	3	1	3

	0	1	2	3	4	5	6	7	8	9	10	11
values	2.0	3.5	6.7	8.2	9.2	1.1	2.8	3.0	1.5	4.5	2.5	8.9

On fait l'hypothèse (raisonnable) que :
`sizeof(double) = 2 sizeof(int).`

- Algorithme : Mémoire $= 37/2$ doubles ; Flops $= 13 \Longrightarrow$ $I_a \simeq 0.7$.
- Machine : Mémoire : 8.73 Giga doubles/s

$$=> \text{Atteignable} = 0.7 \times 8.73 = 6.11 \text{ Gflops}.$$

$$\text{Mesuré} = 6.42 \text{ Gflops}.$$

Intensité arithmétique : expériences

Appiquer le stencil du laplacien en 3-d (7 points), stocké en matrice CSR :

	0	1	2	3	4
0	2.0		3.5		6.7
1		8.2		9.2	
2		1.1	2.8		
3	3.0		1.5	4.5	
4		2.5		8.9	

	0	1	2	3	4	5
rowptr	0	3	5	7	10	12

	0	1	2	3	4	5	6	7	8	9	10	11
colind	0	2	4	1	3	1	2	0	2	3	1	3

	0	1	2	3	4	5	6	7	8	9	10	11
values	2.0	3.5	6.7	8.2	9.2	1.1	2.8	3.0	1.5	4.5	2.5	8.9

On fait l'hypothèse (raisonnable) que :
`sizeof(double) = 2 sizeof(int)`.

- Algorithme : Mémoire $= 37/2$ doubles ; Flops $= 13 \Longrightarrow$ $I_a \simeq 0.7$.
- Machine : Mémoire : 8.73 Giga doubles/s

$$=> \text{Atteignable} = 0.7 \times 8.73 = 6.11 \text{ Gflops}.$$

$$\text{Mesuré} = 6.42 \text{ Gflops}.$$

Note : borné à $0.87 \times 8.73 \simeq 7.6$ Gflops/s quelque soit la structure de données.

Comment connaître la bande passante d'une machine ?

Sur le web

En allant sur le site du constructeur : `ark.intel.com`, `www.amd.com`, ...

En la mesurant

- Divers programmes, dont `stream` `https://www.cs.virginia.edu/stream/`
- Programme en langage bas niveau (`C/C++`, ...).

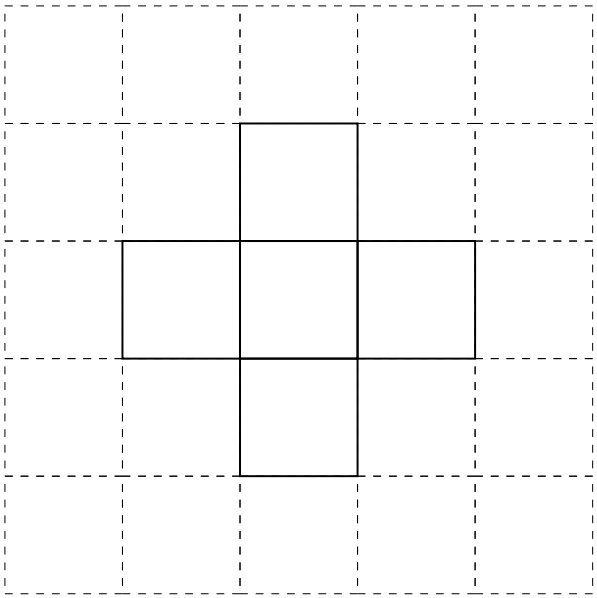
NB : augmenter le plus possible la taille des tableaux !
NB : la bande passante n'est parfois pas atteignable sur un seul cœur $\Rightarrow$ parallélisme.

Est-ce sans espoir ?

On peut (il faut !) favoriser la réutilisation des données stockées dans le cache L1.

Est-ce sans espoir ?

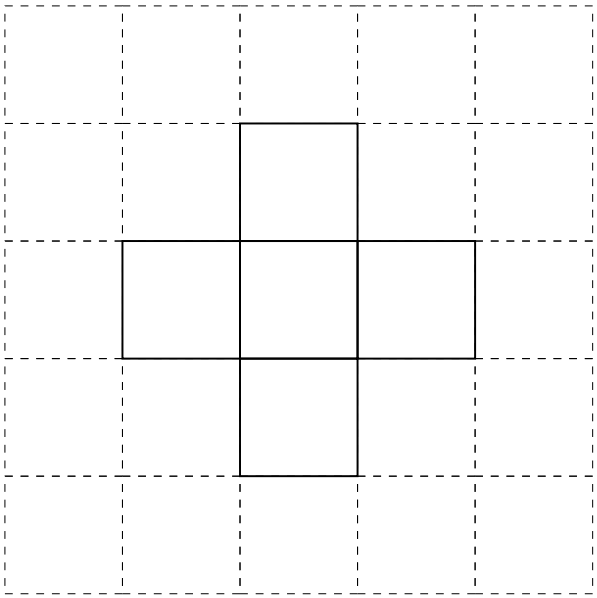
On peut (il faut !) favoriser la réutilisation des données stockées dans le cache L1.

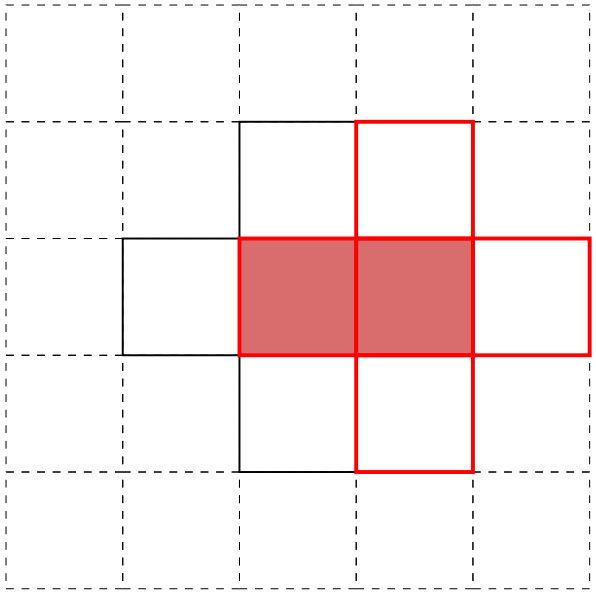


$$u_{ij} = 0.25\ (u_{i+1,j} + u_{i-1,j} - 4u_{ij} + u_{i,j+1} + u_{i,j-1}).$$

Est-ce sans espoir ?

On peut (il faut !) favoriser la réutilisation des données stockées dans le cache L1.





$$u_{ij} = 0.25\ (u_{i+1,j} + u_{i-1,j} - 4u_{ij} + u_{i,j+1} + u_{i,j-1}).$$

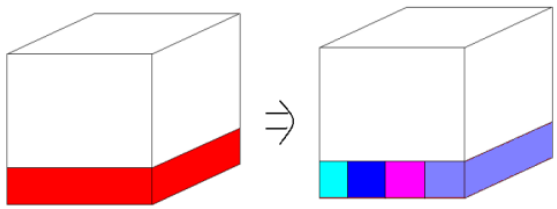
$$Y = \mathrm{Stencil}_7(\mathrm{U}) + \sum_{\mathrm{i}=0}^{\mathrm{k}} \alpha_{\mathrm{i}} \mathrm{V}_{\mathrm{i}}.$$

k	$Max.I_a$	Gflops/s
0	4.0	34.8
1	3.3	29.0
2	3.0	26.1
3	2.8	24.4

$$Y = \mathrm{Stencil}_7(\mathrm{U}) + \sum_{\mathrm{i}=0}^{\mathrm{k}} \alpha_{\mathrm{i}} \mathrm{V}_{\mathrm{i}}.$$

k	$Max.I_a$	Gflops/s
0	4.0	34.8
1	3.3	29.0
2	3.0	26.1
3	2.8	24.4

Programmez les boucles de telle sorte à maximiser la réutilisation des données.



Coupez le domaine en *frites*.

À éviter

- produits scalaires,
- matrices creuses (préconditionnement LU incomplet),
- combinaisons linéaires de grands vecteurs,
- méthodes avec un parallélisme limité,
- ... méthodes avec une faible intensité.

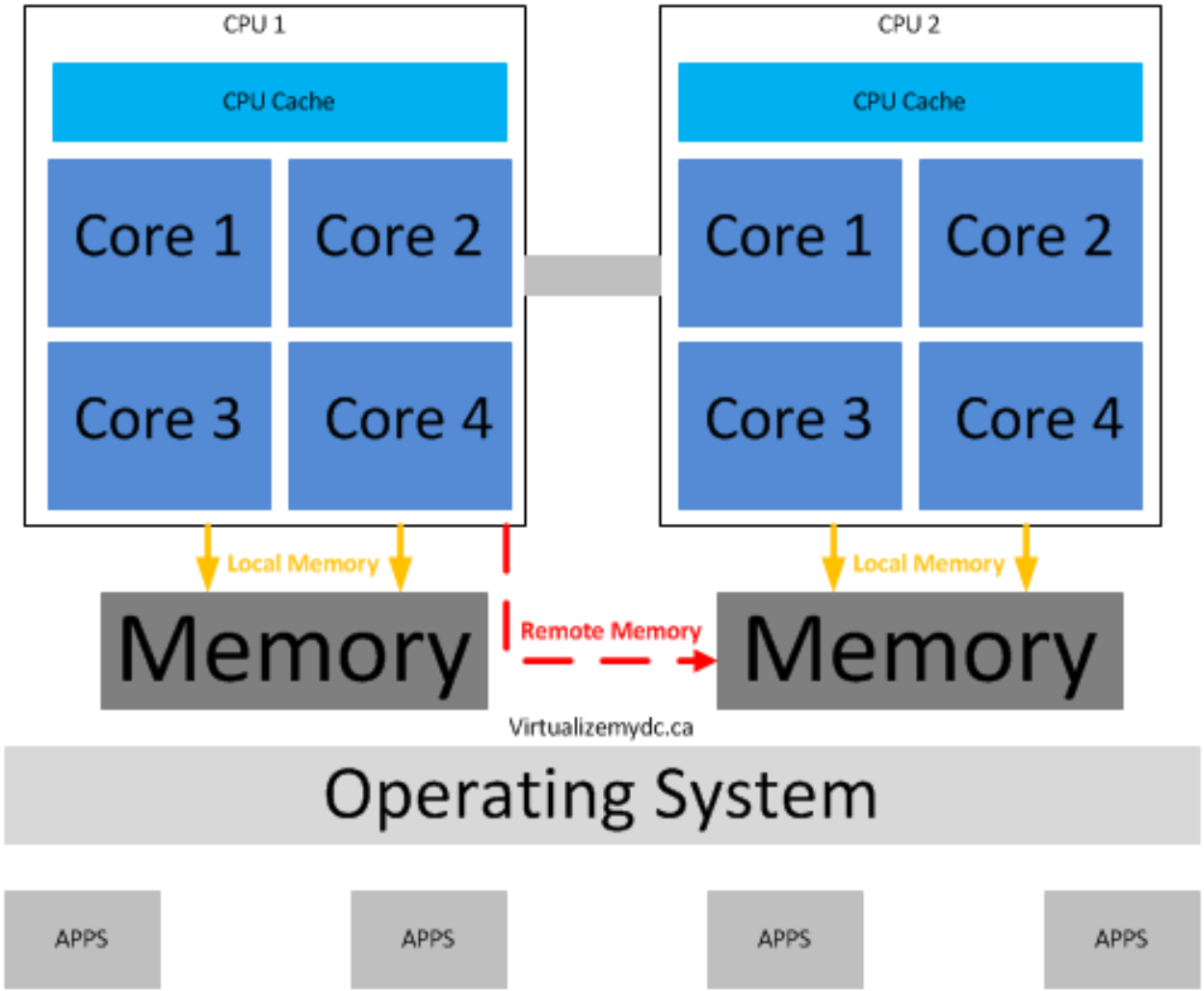
À éviter

- produits scalaires,
- matrices creuses (préconditionnement LU incomplet),
- combinaisons linéaires de grands vecteurs,
- méthodes avec un parallélisme limité,
- ... méthodes avec une faible intensité.

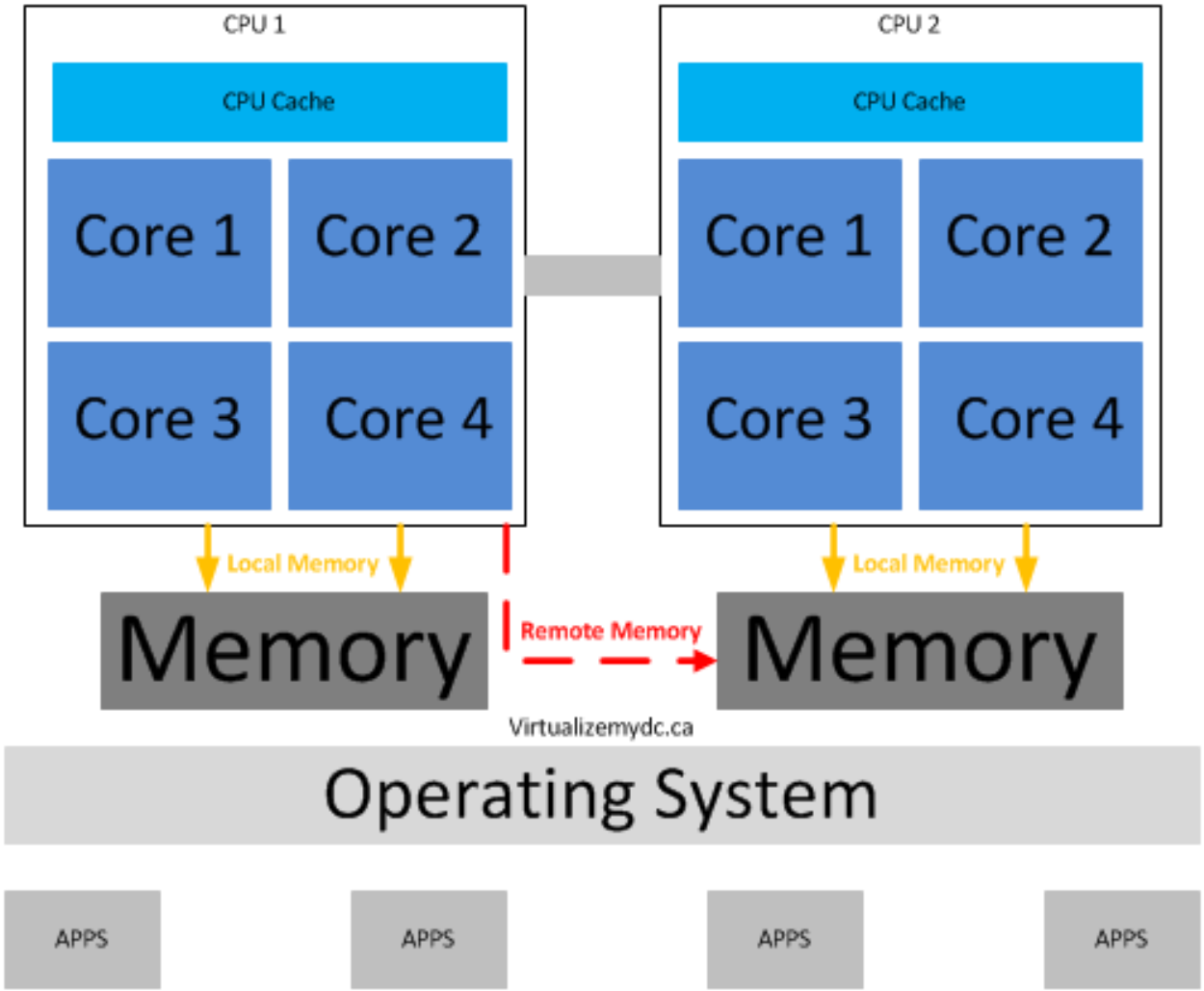
À préférer

- ... méthodes de grande intensité,
- méthodes "embarrassingly" parallèles.

Non Uniform Memory Access (NUMA)



Non Uniform Memory Access (NUMA)



Les accès distants sont très lents !

NUMA : remèdes

Fixer les threads sur les cœurs : `tasket` ou `numactl`

On évite qu'un thread ne migre vers un autre CPU et se retrouve alors à faire des accès distants :

```
$ numactl --cpunodebind=0 --membind=0 mysimulation
```

NUMA : remèdes

Fixer les threads sur les cœurs : `tasket` ou `numactl`

On évite qu'un thread ne migre vers un autre CPU et se retrouve alors à faire des accès distants :

```
$ numactl --cpunodebind=0 --membind=0 mysimulation
```

"Toucher" les données

La mémoire n'est réellement assignée, que lors de la première écriture, par page et en fonction du CPU.

1. allouer la matrice ;
2. créer *n* threads ;
3. initialiser la matrice par bloc et en // (1 thread = 1 bloc) ;
4. faire les opérations par blocs (1 thread = le même bloc).