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

Thierry Dumont

Institut Camille Jordan

Février 2017

Le parallélisme en mémoire partagée expliqué en 2 minutes.

Machines actuelles :

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

Bientôt : une centaine de cœurs par machine.

Le parallélisme en mémoire partagée expliqué en 2 minutes.

Machines actuelles :

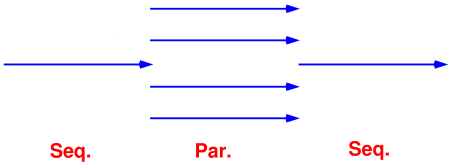
- $1, 2, 4...$ processeurs,
- chaque processeur a n cœurs, avec $n = 2, 4, 8, ...$

Bientôt : une centaine de cœurs par machine.

Parallélisme obligatoire !

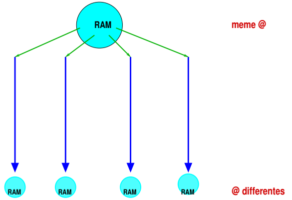
Le parallélisme en mémoire partagée expliqué en 2 minutes.

Processus légers (threads) :



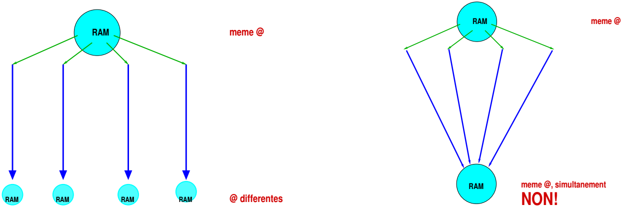
Le parallélisme en mémoire partagée expliqué en 2 minutes.

Partage de la mémoire (sans sécurité).



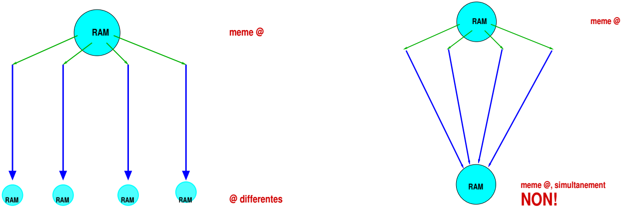
Le parallélisme en mémoire partagée expliqué en 2 minutes.

Partage de la mémoire (sans sécurité).



Le parallélisme en mémoire partagée expliqué en 2 minutes.

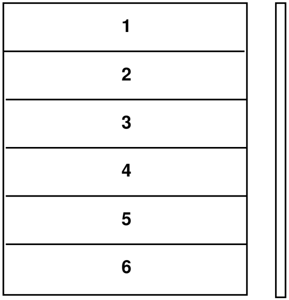
Partage de la mémoire (sans sécurité).



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

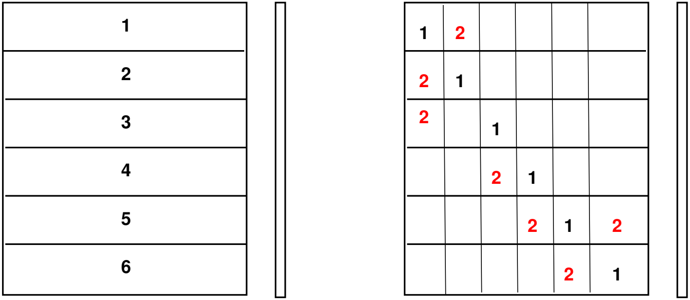
Le parallélisme en mémoire partagée expliqué en 2 minutes.

Exemple : le produit matrice x vecteur.

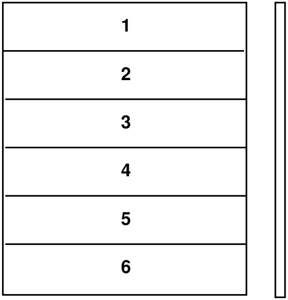


Le parallélisme en mémoire partagée expliqué en 2 minutes.

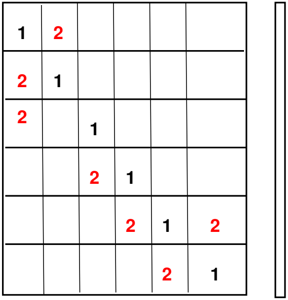
Exemple : le produit matrice x vecteur.



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) !

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/second).

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/second).
Aspect SIMD :

AVX instructions :

In one clock cycle, do :
$y_i = \; a_i \, x_i \; + \; b_i, \;\; i = 1, 4$ (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/second).
Aspect SIMD :

AVX instructions :

In one clock cycle, do :
$y_i = \; a_i \, x_i \; + \; b_i, \; i = 1,4$ (8 flops).

La performance *peak* est donc de :

2×8

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/second).
Aspect SIMD :

AVX instructions :

In one clock cycle, do :
$y_i = \ a_i \ x_i \ + \ b_i, \ i = 1, 4$ (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/second).
Aspect SIMD :

AVX instructions :

In one clock cycle, do :
$y_i = \ a_i \ x_i \ + \ b_i, \ i = 1, 4$ (8 flops).

La performance *peak* est donc de :

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

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/second).
Aspect SIMD :

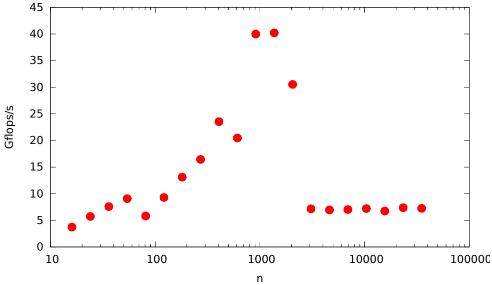
AVX instructions :

In one clock cycle, do :
$y_i = \ a_i \ x_i \ + \ b_i, \ i = 1, 4$ (8 flops).

La performance *peak* est donc de :

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

Blas Intel, parallèle.



Compte 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.**

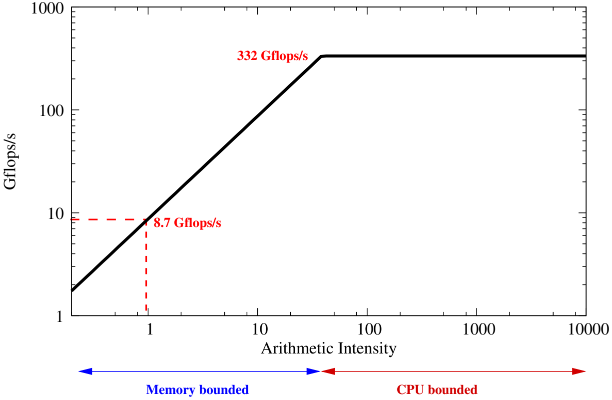
Intensité arithmétique

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

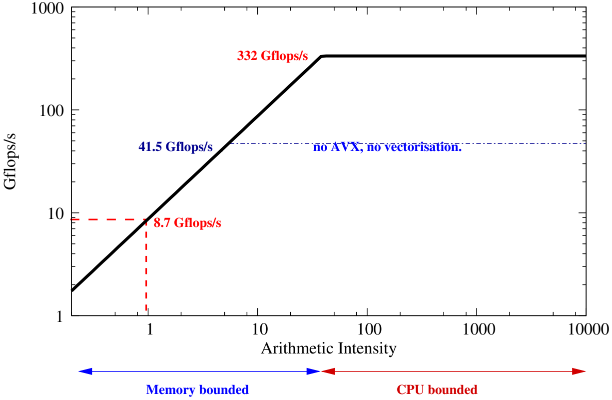
GFlops/sec atteignables $=$

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

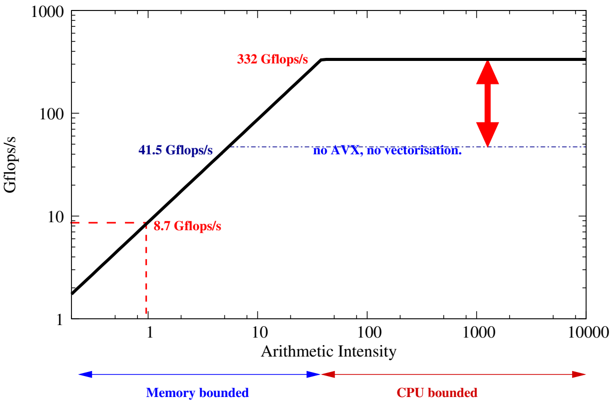
Le « Roofline Model »



Le « Roofline Model »



Le « Roofline Model »



Williams S. et al: *Roofline : An Insightful Visual Performance Model for Multicore Architectures* – Commun. ACM, 1999.

Intensité arithmétique et Roofline Model : quelques exemples

Unitée utilisée : double.

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

Intensité arithmétique et Roofline Model : quelques exemples

Unitée 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

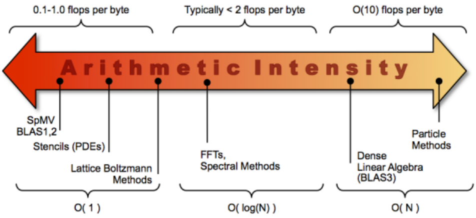
Unitée utilisée : double.

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 $\times$ Matrix $C = A.B$: $I_a = 2\ n^3/4\ n^2 = \mathcal{O}(n)$.

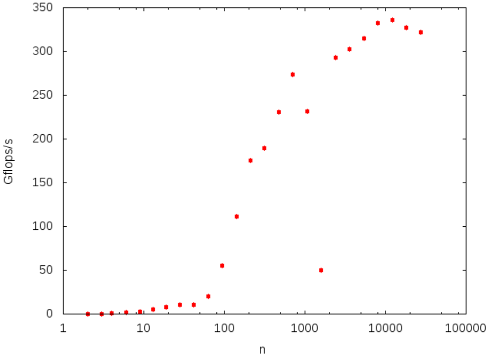
Intensité arithmétique et Roofline Model : quelques exemples

Unitée 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 $\times$ Matrix $C = A.B$: $I_a = 2\ n^3/4\ n^2 = \mathcal{O}(n)$.

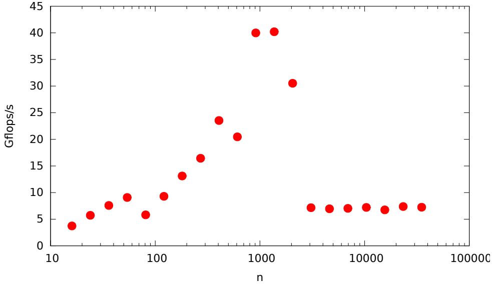


Matrix $\times$ Matrix product (DGEMM, Intel mkl parallel version) :



Intensité arithmétique : experiences

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



Intensité arithmétique : experiences

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

Intensité arithmétique : experiences

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 : experiences

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 (raisonable) que :
`sizeof(double) = 2 sizeof(int).`

- Algorithm Bdwth : $37/2$ double ; Flops : 13 $=>$ $I_a \simeq 0.7$.

Intensité arithmétique : experiences

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 (raisonable) que :

`sizeof(double) = 2 sizeof(int).`

- Algorithm Bdwth : 37/2 double ; Flops : 13 => $I_a \simeq 0.7$.
- Machine Bdwth : 8.73 Giga doubles/s

=> Attainable $= 0.7 \times 8.73 = 6.11$ Gflops.

Intensité arithmétique : experiences

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 (raisonable) que :

`sizeof(double) = 2 sizeof(int).`

- Algorithm Bdwth : $37/2$ double ; Flops : 13 $=>$ $I_a \simeq 0.7$.
- Machine Bdwth : 8.73 Giga doubles/s

$=>$ Attainable $= 0.7 \times 8.73 = 6.11$ Gflops.

Measured : 6.42 Gflops.

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 (raisonable) que :
`sizeof(double) = 2 sizeof(int).`

- Algorithm Bdwth : $37/2$ double ; Flops : 13 $=>$ $I_a \simeq 0.7$.
- Machine Bdwth : 8.73 Giga doubles/s

$=>$ Attainable $= 0.7 \times 8.73 = 6.11$ Gflops.

Measured : 6.42 Gflops.

Note : bounded to $1.15 \times 8.73 \simeq 10$ Gflops/s whatever the data structure.

Comment mesurer la bande passante d'une machine ?

Divers programmes, dont `stream`

`https://www.cs.virginia.edu/stream/`

Programme en C.

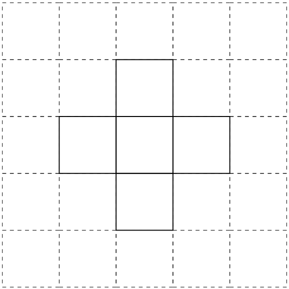
NB : augmenter le plus possible la taille des tableaux !

Est-ce sans espoir ?

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

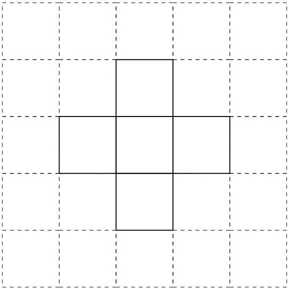
Est-ce sans espoir ?

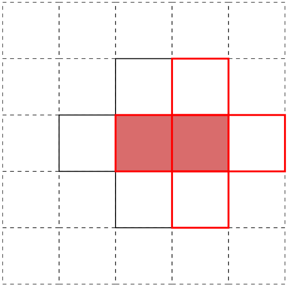
On peut (il faut !) favoriser la réutilisation des données stockées dans le cache L1.



Est-ce sans espoir ?

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





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

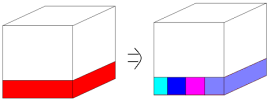
$$Y = \mathrm{Stencil}_7(\mathrm{U}) + \sum_{\mathrm{i}=0}^{\mathrm{k}} \alpha_{\mathrm{i}} \mathrm{V}_{\mathrm{i}}.$$

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

Program the loops so as to maximize data re-use.



Cut the domain in *French Fries*.

Avoid :

- dot products,
- sparse matrices (incomplete LU preconditioning),
- linear combination of large vectors.
- methods with a limited parallelism.
- ... low intensity methods.

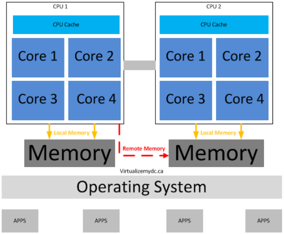
Avoid :

- dot products,
- sparse matrices (incomplete LU preconditioning),
- linear combination of large vectors.
- methods with a limited parallelism.
- ... low intensity methods.

Prefer :

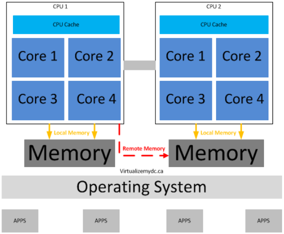
- ... high intensity methods,
- embarrassingly parallel methods.

Non Uniform Memory Access (NUMA)



Les accès « remote »sont très lents !

Non Uniform Memory Access (NUMA)



Les accès « remote »sont très lents !

Remèdes :

1. fixer les threads sur les cœurs.
2. « touch »des données.