

Complexité, intensité arithmétique et Roofline model

Roland DENIS

CNRS / Institut Camille Jordan

8 avril 2021

Complexité algorithmique

Complexité des algorithmes

Lors de l'écriture de codes, il est important d'analyser leur **complexité** qui permet d'en estimer les performances et de le comparer à d'autres algorithmes.

Complexité

La complexité correspond à une **comparaison asymptotique** d'une **mesure de performance** (le temps d'exécution ou la mémoire utilisée) en fonction de **paramètres caractéristiques** de l'algo (la dimension de l'espace, la taille du vecteur, ...).

Complexité des algorithmes

Lors de l'écriture de codes, il est important d'analyser leur **complexité** qui permet d'en estimer les performances et de le comparer à d'autres algorithmes.

Complexité

La complexité correspond à une **comparaison asymptotique** d'une **mesure de performance** (le temps d'exécution ou la mémoire utilisée) en fonction de **paramètres caractéristiques** de l'algo (la dimension de l'espace, la taille du vecteur, ...).

Types de complexité

On parle de complexité :

- dans le **meilleur des cas** quand les données fournies en entrée sont les **plus favorables** à l'algorithme,
- dans le **pire des cas** quand les données fournies en entrée sont les **plus défavorables** à l'algorithme,
- **moyenne** pour la performance moyenne étant donnée une **distribution** sur l'ensemble des données en entrée possibles.

Exemples de complexités en temps

- accès au n -ième élément d'un vecteur : complexité **constante**, en $\mathcal{O}(1)$,
- recherche par dichotomie dans un vecteur de taille N : complexité **logarithmique**, en $\mathcal{O}(\log(N))$,
- calcul de la norme d'un vecteur de taille N : complexité **linéaire**, en $\mathcal{O}(N)$,
- tri d'un vecteur de taille N par la méthode du tri fusion : complexité **linéarithmétique**, en $\mathcal{O}(N \log(N))$,
- multiplication matrice \times vecteur de taille N : complexité **quadratique** (polynomiale), en $\mathcal{O}(N^2)$,
- méthode du pivot de Gauss en dimension N : complexité **cubique** (polynomiale), en $\mathcal{O}(N^3)$,
- problème du rangement de N objets dans un sac à dos, par force brute : complexité **exponentielle** en $\mathcal{O}(N2^N)$.

Voir fr.wikipedia.org/wiki/Analyse_de_la_complexité_des_algorithmes

Produit scalaire

Pour le produit scalaire de deux vecteurs de tailles n , on réalise n produits et $n - 1$ sommes, pour obtenir un scalaire.

On a donc une complexité $\mathcal{O}(n)$ en temps et $\mathcal{O}(1)$ en espace.

Exemples

Produit scalaire

Pour le produit scalaire de deux vecteurs de tailles n , on réalise n produits et $n - 1$ sommes, pour obtenir un scalaire.

On a donc une complexité $\mathcal{O}(n)$ en temps et $\mathcal{O}(1)$ en espace.

Produit matrice \times vecteur

Pour le produit d'une matrice de taille $m \times n$ par un vecteur de taille n , on réalise n produits et $n - 1$ sommes pour chaque élément du vecteur final de taille m .

On a donc une complexité $\mathcal{O}(n \times m)$ en temps et $\mathcal{O}(m)$ en espace.

Exemples

Produit scalaire

Pour le produit scalaire de deux vecteurs de tailles n , on réalise n produits et $n - 1$ sommes, pour obtenir un scalaire.

On a donc une complexité $\mathcal{O}(n)$ en temps et $\mathcal{O}(1)$ en espace.

Produit matrice \times vecteur

Pour le produit d'une matrice de taille $m \times n$ par un vecteur de taille n , on réalise n produits et $n - 1$ sommes pour chaque élément du vecteur final de taille m .

On a donc une complexité $\mathcal{O}(n \times m)$ en temps et $\mathcal{O}(m)$ en espace.

Produit matrice \times matrice

Pour le produit de deux matrices de taille $m \times n$ et $n \times p$, on réalise n produits + n sommes pour chaque élément de la matrice finale de taille $m \times p$.

On a donc une complexité $\mathcal{O}(n \times m \times p)$ en temps et $\mathcal{O}(m \times p)$ en espace.

Pour le cas où $m = p = n$, on a $\mathcal{O}(n^3)$ en temps et $\mathcal{O}(n^2)$ en espace.

C'est l'approche naïve, la meilleure complexité à ce jour étant en $\mathcal{O}(n^{2.3728639})$ (see https://en.wikipedia.org/wiki/Matrix_multiplication).

Intensité arithmétique

Pourquoi je n'obtiens pas les performances attendues ?

Un calcul ne peut être effectué que si :

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

Pourquoi je n'obtiens pas les performances attendues ?

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

L'**intensité arithmétique** prend en compte ce facteur limitant et permet de mieux comprendre le comportement d'un noyau de calcul par rapport à une architecture cible :

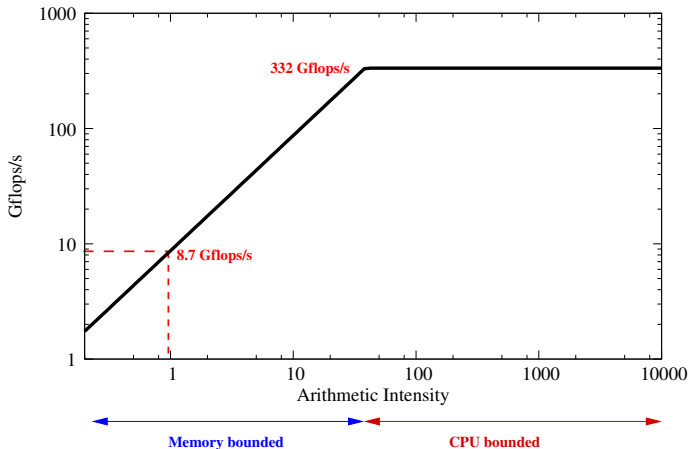
$$I_a = \frac{\text{nombre d'opérations}}{\text{quantité de mémoire échangée}}$$

On peut donc l'augmenter en favorisant le **réemploi des données**, via par exemple une utilisation efficace des différents niveaux de **cache** du processeur.

Le « Roofline Model »

Performance [GFlops/sec] atteignable (*Peak performance*)

$\min(\text{Performance cr\^ete}, \text{Bande passante m\^emoire} \cdot I_a)$



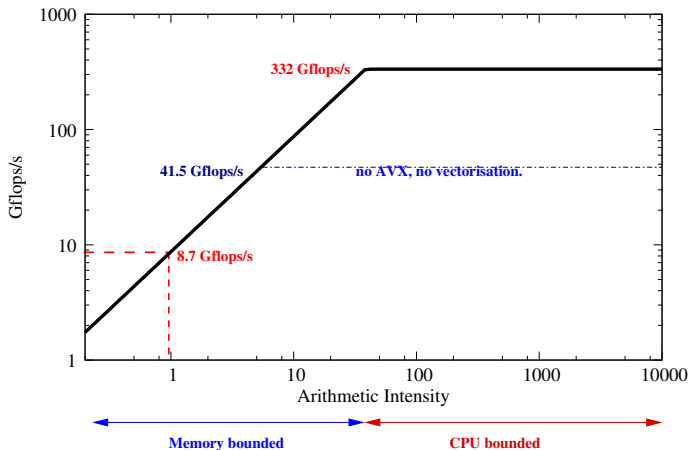
Williams S. et al: *Roofline : An Insightful Visual Performance Model for Multicore Architectures*

Compass, ACM, 2009

Le « Roofline Model »

Performance [GFlops/sec] atteignable (*Peak performance*)

$\min(\text{Performance cr\^ete}, \text{Bande passante m\^emoire} \cdot I_a)$



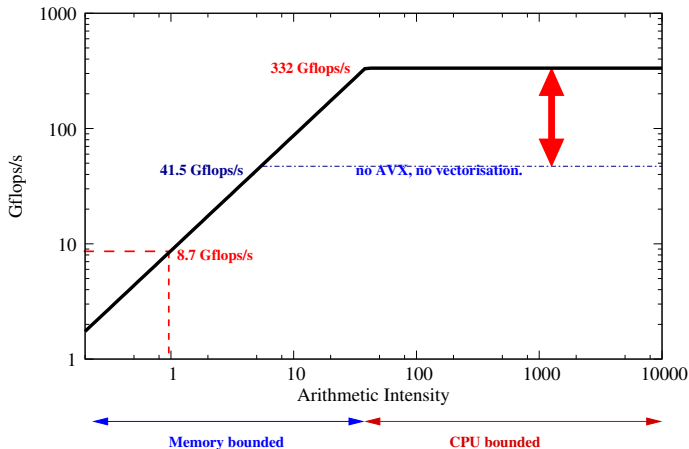
Williams S. et al: *Roofline : An Insightful Visual Performance Model for Multicore Architectures*

Compassus, ACM, 2009

Le « Roofline Model »

Performance [GFlops/sec] atteignable (*Peak performance*)

$\min(\text{Performance cr\^ete}, \text{Bande passante m\^emoire} \cdot I_a)$



Williams S. et al: *Roofline : An Insightful Visual Performance Model for Multicore Architectures*

Compassus, ACM, 2009

Produit scalaire

Pour le produit scalaire de deux vecteurs de tailles n , on réalise n produits et $n - 1$ sommes, en lisant/écrivant $2n + 1$ valeurs.

On a donc $I_a = \mathcal{O}(1)$.

Intensité arithmétiques : quelques exemples

Produit scalaire

Pour le produit scalaire de deux vecteurs de tailles n , on réalise n produits et $n - 1$ sommes, en lisant/écrivant $2n + 1$ valeurs.

On a donc $I_a = \mathcal{O}(1)$.

Produit matrice \times vecteur

Pour le produit d'une matrice de taille $n \times n$ par un vecteur de taille n , on réalise n produits et $n - 1$ sommes pour chaque élément du vecteur final de taille n , en lisant/écrivant $n^2 + 2n$ valeurs.

On a donc une intensité arithmétique de $I_a = \frac{n(2n-1)}{n^2+2n} = \mathcal{O}(1)$.

Intensité arithmétiques : quelques exemples

Produit scalaire

Pour le produit scalaire de deux vecteurs de tailles n , on réalise n produits et $n - 1$ sommes, en lisant/écrivant $2n + 1$ valeurs.

On a donc $I_a = \mathcal{O}(1)$.

Produit matrice \times vecteur

Pour le produit d'une matrice de taille $n \times n$ par un vecteur de taille n , on réalise n produits et $n - 1$ sommes pour chaque élément du vecteur final de taille n , en lisant/écrivant $n^2 + 2n$ valeurs.

On a donc une intensité arithmétique de $I_a = \frac{n(2n-1)}{n^2+2n} = \mathcal{O}(1)$.

Produit matrice \times matrice

Pour le produit de deux matrices de taille $n \times n$, on réalise n produits $+ n - 1$ sommes pour chaque élément de la matrice finale de taille $n \times n$, en lisant/écrivant $3n^2$ valeurs.

On a donc une intensité arithmétique de $I_a = \frac{n^2(2n-1)}{3n^2} = \mathcal{O}(n)$.

Laplacien 3D en différences finies

Appliquer le stencil du Laplacien à 7 points en dimension 3 :

$$\Delta u(x, y, z) \approx \frac{u(x-h, y, z) + u(x+h, y, z) + \dots - 6u(x, y, z)}{h^2}$$

d'où une intensité arithmétique en $I_a = 8/8 = 1$.

Laplacien 3D en différences finies

Appliquer le stencil du Laplacien à 7 points en dimension 3 :

$$\Delta u(x, y, z) \approx \frac{u(x-h, y, z) + u(x+h, y, z) + \dots - 6u(x, y, z)}{h^2}$$

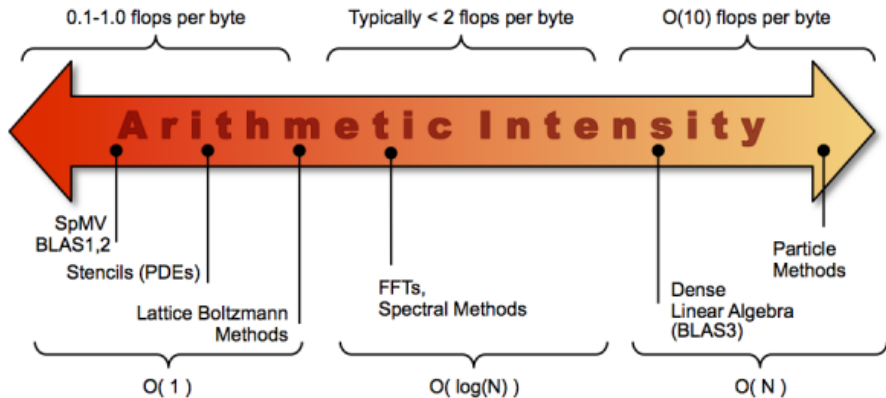
d'où une intensité arithmétique en $I_a = 8/8 = 1$.

Cache mémoire

Attention : une mauvaise utilisation du cache mémoire peut dégrader l'intensité arithmétique (matrice \times matrice en $\mathcal{O}(1)$).

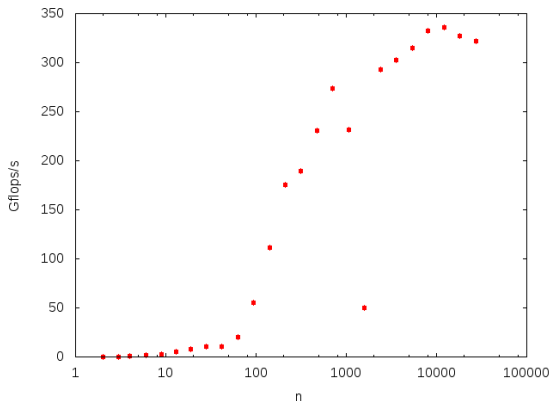
Intensité arithmétiques : quelques exemples

Quelques méthodes numériques courantes



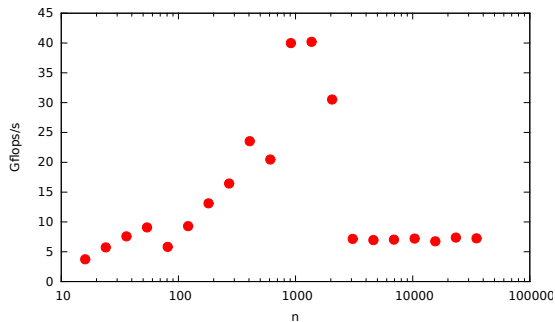
Intensité arithmétique : expériences

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



Intensité arithmétique : expériences

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



Intensité arithmétique : expériences

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

Intensité arithmétique : expériences

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

	0	1	2	3	4		0	1	2	3	4	5						
0	2.0		3.5		6.7	rowptr	0	3	5	7	10	12						
1		8.2		9.2			0	1	2	3	4	5	6	7	8	9	10	11
2		1.1	2.8			colind	0	2	4	1	3	1	2	0	2	3	1	3
3	3.0		1.5	4.5			0	1	2	3	4	5	6	7	8	9	10	11
4		2.5		8.9		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

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

	0	1	2	3	4		0	1	2	3	4	5						
0	2.0		3.5		6.7	rowptr	0	3	5	7	10	12						
1		8.2		9.2			0	1	2	3	4	5	6	7	8	9	10	11
2		1.1	2.8			colind	0	2	4	1	3	1	2	0	2	3	1	3
3	3.0		1.5	4.5			0	1	2	3	4	5	6	7	8	9	10	11
4		2.5		8.9		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 = 36/2 doubles; Flops = 13 $\implies I_a \simeq 0.722$.

Intensité arithmétique : expériences

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

	0	1	2	3	4		0	1	2	3	4	5						
	2.0		3.5		6.7	rowptr	0	3	5	7	10	12						
1		8.2		9.2			0	1	2	3	4	5	6	7	8	9	10	11
2		1.1	2.8			colind	0	2	4	1	3	1	2	0	2	3	1	3
3	3.0		1.5	4.5			0	1	2	3	4	5	6	7	8	9	10	11
4		2.5		8.9		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 = 36/2 doubles; Flops = 13 $\implies I_a \simeq 0.722$.
- Machine : Mémoire : 8.73 Giga doubles/s
 \implies Atteignable = $0.722 \times 8.73 = 6.30$ Gflops.

Intensité arithmétique : expériences

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

	0	1	2	3	4		0	1	2	3	4	5						
	2.0		3.5		6.7	rowptr	0	3	5	7	10	12						
1		8.2		9.2			0	1	2	3	4	5	6	7	8	9	10	11
2		1.1	2.8			colind	0	2	4	1	3	1	2	0	2	3	1	3
3	3.0		1.5	4.5			0	1	2	3	4	5	6	7	8	9	10	11
4		2.5		8.9		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 = 36/2 doubles; Flops = 13 $\implies I_a \simeq 0.722$.
- Machine : Mémoire : 8.73 Giga doubles/s
 \implies Atteignable = $0.722 \times 8.73 = 6.30$ Gflops.

Mesuré = 6.42 Gflops.

Intensité arithmétique : expériences

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

	0	1	2	3	4		0	1	2	3	4	5						
0	2.0		3.5		6.7	rowptr	0	3	5	7	10	12						
1		8.2		9.2			0	1	2	3	4	5	6	7	8	9	10	11
2		1.1	2.8			colind	0	2	4	1	3	1	2	0	2	3	1	3
3	3.0		1.5	4.5			0	1	2	3	4	5	6	7	8	9	10	11
4		2.5		8.9		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 = 36/2 doubles; Flops = 13 $\implies I_a \simeq 0.722$.
- Machine : Mémoire : 8.73 Giga doubles/s
 \implies Atteignable = $0.722 \times 8.73 = 6.30$ Gflops.

Mesuré = 6.42 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 !

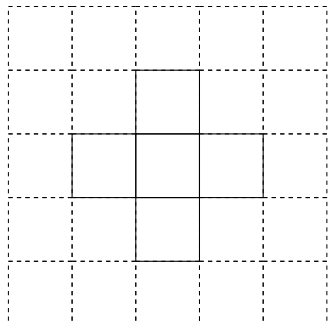
NB : la bande passante n'est parfois pas atteignable sur un seul cœur ⇒ parallélisme.

Est-ce sans espoir ?

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

Est-ce sans espoir ?

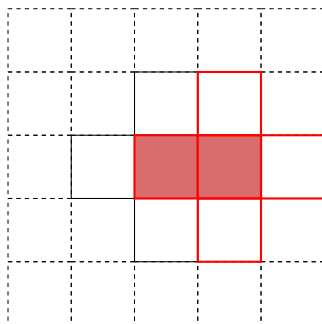
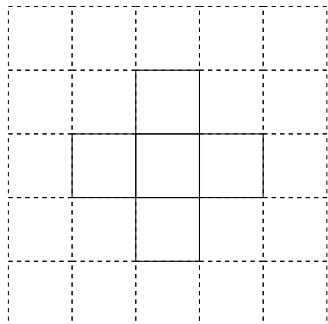
On peut (il faut !) favoriser la réutilisation des données stockées dans le cache L1.



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

Est-ce sans espoir ?

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



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