Projet - LU2IN013

Vers des implémantations haute performance pour le calcul matriciel

LIN Clément - XIAO Carine - XU Audic

Sorbonne Université

2021 - 2022

Sommaire

Introduction

Le sujet

- Algorithme et calcul haute performance pour l'algèbre linéaire
- Le calcul matriciel

Introduction

Le sujet

- Algorithme et calcul haute performance pour l'algèbre linéaire
- Le calcul matriciel

Domaines d'applications

- Climatologie
- Biologie
- Imagerie
- Cryptologie
- ...

Introduction

Le sujet

- Algorithme et calcul haute performance pour l'algèbre linéaire
- Le calcul matriciel

Domaines d'applications

- Climatologie
- Biologie
- Imagerie
- Cryptologie
- ...

L'objectif

- Analyses d'algorithmes naïfs
- Implémentations d'optimisations

Bibliothèque FLINT

1. Langage : *C*
2. Première publication : 2007
3. Principaux contributeurs : David Harvey et William Hart
4. Dernière version : 24/06/2022
5. github.com/wbhart/flint2

Bibliothèques externes

FLINT et NTL

Bibliothèque FLINT

1. Langage : C
2. Première publication : 2007
3. Principaux contributeurs : David Harvey et William Hart
4. Dernière version : 24/06/2022
5. github.com/wbhart/flint2

Bibliothèque NTL

1. Langage : $C++$
2. Première publication : 1990
3. Principal contributeur : Victor Shoup
4. Dernière version : 23/06/2022
5. github.com/libntl/ntl

Bibliothèques pour la théorie des nombres : FLINT et NTL.

Version	Naïf	FLINT	NTL
Langage	C	C	C++
Modulo	Oui	Oui	Oui
Coefficient	int_32	mp_limb_t	zz_p

Premières comparaisons

Taille	Naïf (en s)	FLINT (en s)	NTL (en s)
16	0.00014	0.00001	0.00001
64	0.00792	0.00027	0.00023
256	0.44694	0.01239	0.01087
512	3.55452	0.07720	0.06635
1024	28.46430	0.54735	0.46831
4096	1909.08744	27.32232	23.48177

En théorie ...

Soient A et B de taille n $\in \mathbb{N}$ à coefficients int_32.

1. $A \times B$: $2n^3 - n^2$ opérations
2. 2 000 000 000 opérations : 3.21 secondes

Pour des matrices de tailles 1024, on a 2 146 435 072 opérations à faire, ce qui devrait prendre 3.44 secondes.

En théorie ...

Soient A et B de taille n $\in \mathbb{N}$ à coefficients int_32.

1. $A \times B$: $2n^3 - n^2$ opérations
2. 2 000 000 000 opérations : 3.21 secondes

Pour des matrices de tailles 1024, on a 2 146 435 072 opérations à faire, ce qui devrait prendre 3.44 secondes.

Dans la pratique ...

Deux matrices de tailles 1024 à coefficients int_32 prend 28.5 secondes !

En théorie ...

Soient A et B de taille n $\in \mathbb{N}$ à coefficients int_32.

1. $A \times B$: $2n^3 - n^2$ opérations
2. 2 000 000 000 opérations : 3.21 secondes

Pour des matrices de tailles 1024, on a 2 146 435 072 opérations à faire, ce qui devrait prendre 3.44 secondes.

Dans la pratique ...

Deux matrices de tailles 1024 à coefficients int_32 prend 28.5 secondes !

À quoi est dûe cette différence ?

Quel est le coût des déplacements mémoires ?

Quel est le coût des déplacements mémoires ?

En les comparant :

Taille	Int 32 sans	Int 32 avec	Int 64 sans	Int 64 avec
256	0.33267	0.44694	0.21368	0.22477
512	3.04866	3.55452	1.68624	1.96220
1024	25.27146	28.46430	13.35585	15.83425
2048	204.4302	225.7763	107.78951	216.7815

Quel est le coût des déplacements mémoires ?

En les comparant :

Taille	Int 32 sans	Int 32 avec	Int 64 sans	Int 64 avec
256	0.33267	0.44694	0.21368	0.22477
512	3.04866	3.55452	1.68624	1.96220
1024	25.27146	28.46430	13.35585	15.83425
2048	204.4302	225.7763	107.78951	216.7815

L'impact de la localité des données est visible et devrait être utilisé intelligemment.

Quel est le coût du modulo ?

Quel est le coût du modulo ?

Taille	Int 32 sans	Int 32 avec	Int 64 sans	Int 64 avec
64	0.00014	0.00648	0.00148	0.00385
128	0.01049	0.04418	0.01016	0.02555
256	0.07631	0.44694	0.08534	0.22477
512	0.79295	3.55452	1.01459	1.96220
1024	7.36199	28.46430	11.62199	15.83425
2048	100.2976	225.7763	158.11522	216.7815

Définition

Le principe de localité des données se décompose en deux étapes :

1. Localité temporelle
2. Localité spatiale

Définition

Le principe de localité des données se décompose en deux étapes :

1. Localité temporelle
2. Localité spatiale

Définition

Le cache est une zone mémoire à l'intérieur du CPU d'accès rapide.

Produit matriciel par transposée

Soient $A = (a_{i,j})_{1 \leq i,j \leq n}, B = (b_{i,j})_{1 \leq i,j \leq n}$ et $C = (c_{i,j})_{1 \leq i,j \leq n} \in \mathcal{M}_n(\frac{\mathbb{Z}}{p\mathbb{Z}})$ telles que $A \times B = C$. Le produit matriciel par transposée est donné par :

$$\forall i,j \in [\![1,n]\!], c_{(i,j)} = (\sum_{k=1}^{n} (a_{(i,k)} \times {}^t b_{(j,k)} \mod p) \mod p)$$

Produit matriciel par blocs

Soient A, B et $C \in \mathcal{M}_n(\frac{\mathbb{Z}}{p\mathbb{Z}})$ telles que $A \times B = C$.
Par la division euclidienne, $n = q \times b + r$ où q est le quotient et r le reste.
On peut écrire alors :

$$A = \begin{bmatrix} A_{1,1} & \cdots & A_{1,q} \\ \vdots & \ddots & \vdots \\ A_{q,1} & \cdots & A_{q,q} \end{bmatrix}$$

De même pour $B = (B_{i,j})_{1 \leq i,j \leq q}$ et $C = (C_{i,j})_{1, \leq i,j \leq q}$ Le produit matriciel par blocs :

$$\forall i, j \in [\![1, q]\!], C_{(i,j)} = \sum_{k=1}^{q} A_{(i,k)} \times B_{(k,j)}$$

Modulo

Implémentation de la décomposition

Principe

Réduire le nombre de modulo en décomposant les multiplications. On passe de $n^2(n+1)$ modulo à faire à $3n^2$.

Principe

Réduire le nombre de modulo en décomposant les multiplications. On passe de $n^2(n+1)$ modulo à faire à $3n^2$.

Départ

On a : $\forall (i,j) \in [\![1,n]\!]^2, c_{i,j} = (\sum_{k=1}^{n}((a_{i,k} \times b_{k,j}) \mod p)) \mod p$.

Principe

Réduire le nombre de modulo en décomposant les multiplications. On passe de $n^2(n+1)$ modulo à faire à $3n^2$.

Départ

On a : $\forall (i,j) \in [\![1,n]\!]^2, c_{i,j} = (\sum_{k=1}^{n}((a_{i,k} \times b_{k,j}) \mod p)) \mod p$.

Arrivée

$\forall (i,j) \in [\![1,n]\!]^2$:

$$\begin{aligned} c_{i,j} &= (\sum_{k=1}^{n}((a_{i,k} \times b_{k,j}) \mod p)) \mod p \\ &= (\sum_{k=1}^{n}((h_k \times 2^{32} + l_k) \mod p)) \mod p \\ &= ((h \mod p) \times 2^{32} + (l \mod p)) \mod p. \end{aligned}$$

Comparaison du modulo classique et de la réduction de Barrett.

Modulo Naïf

Soient a et b $\in \mathbb{N}$.

Calcul de $a \mod b$ en C :

1. $q = a/b$
2. $a \mod b = a - q \times b$

Modulo

Implémentation de Barrett

Comparaison du modulo classique et de la réduction de Barrett.

Modulo Naïf

Soient a et b $\in \mathbb{N}$.

Calcul de $a \mod b$ en C :

1. $q = a/b$
2. $a \mod b = a - q \times b$

Attention

La division est une opération lente.

Modulo

Implémentation de Barrett

Comparaison du modulo classique et de la réduction de Barrett.

Modulo Naïf

Soient a et b $\in \mathbb{N}$.

Calcul de $a \mod b$ en C :

1. $q = a/b$
2. $a \mod b = a - q \times b$

Attention

La division est une opération lente.

Réduction de Barrett

Approximation du quotient a/b par : $\lfloor (\lfloor \frac{a}{2^{30}} \rfloor \times \lfloor \frac{2^{62}}{b} \rfloor) \times \frac{1}{2^{32}} \rfloor$.

Remarque : Diviser par une puissance de 2 est "gratuit" en temps

Résultat final

Résultat final

Dernières comparaisons

Taille	Naïf	Final	FLINT	NTL
128	0.02555	0.00219	0.00187	0.00157
256	0.22477	0.01573	0.01239	0.01087
512	1.96220	0.1186	0.07720	0.06635
1024	15.83425	0.98316	0.54735	0.46831
2048	216.78154	12.59745	3.83937	3.31264
4096	1831.51037	169.77569	27.32231	23.48177

Quelques pourcentages représentant le gain de temps

Taille	Final	FLINT	NTL
128	91.4	92.7	93.9
256	93	94.5	95.2
512	94.0	96.1	96.6
1024	93.8	96.5	97

Remerciements

Ce que nous avons appris

1. Travailler en équipe (Git)
2. Latex & Beamer
3. Domaine du calcul haute performance
4. Rigueur, démarche scientifique, etc ..
5. Compétences techniques (installation de bibliothèques, makefile, débogage etc ...)