Mécanismes automatiques d'optimisation des bibliothèques.

Thierry Dumont

Institut Camille Jordan
Université Lyon 1

But: obtenir de bonnes performances sur une grande variété de matériels, sans connaissance précise du matériel

But: obtenir de bonnes performances sur une grande variété de matériels, sans connaissance précise du matériel

Deux exemples:

1. ATLAS (blas),
2. FFTW (transformée de Fourier rapide).

Modèle

- Machine avec mémoire cache,
- minimisation automatique des défauts de cache
- générateurs de code.

Blas

BASIC LINEAR ALGEBRA SUBROUTINES

1. BLAS level 1: opérations sur des vecteurs,
2. BLAS Level 2: opérations (matrices, vecteurs),
3. BLAS Level 3: opération (matrices, matrices).

Blas

BASIC LINEAR ALGEBRA SUBROUTINES

1. BLAS level 1: opérations sur des vecteurs,
2. BLAS Level 2: opérations (matrices, vecteurs),
3. BLAS Level 3: opération (matrices, matrices).

Faire reposer les codes d'algèbre linéaire sur les BLAS: possibilité de bénéficier des optimisations.

optimisations possibles des BLAS

- BLAS 1: faible, le compilateur s'en charge!

optimisations possibles des BLAS

- BLAS 1: faible, le compilateur s'en charge!
- BLAS 2: optimisation entre 10 et 300 %: bonne gestion de la mémoire, réécriture facilitant l'optimisation par le compilateur.

optimisations possibles des BLAS

- BLAS 1: faible, le compilateur s'en charge!
- BLAS 2: optimisation entre 10 et 300 %: bonne gestion de la mémoire, réécriture facilitant l'optimisation par le compilateur.
- BLAS 3: le terrain de jeu!

optimisations possibles des BLAS

- BLAS 1: faible, le compilateur s'en charge!
- BLAS 2: optimisation entre 10 et 300 %: bonne gestion de la mémoire, réécriture facilitant l'optimisation par le compilateur.
- BLAS 3: le terrain de jeu!

Optimisation des produits de matrices

$C = AB,\ C = A^T B,\ C = A^T B + \beta C$ etc...

Optimisation des produits de matrices

$C = AB$, $C = A^T B$, $C = A^T B + \beta C$ etc...
Selon la taille des matrices:

1. copie ou non des matrices dans le cache,

Optimisation des produits de matrices

$C = AB$, $C = A^T B$, $C = A^T B + \beta C$ etc...
Selon la taille des matrices:

1. copie ou non des matrices dans le cache,
2. copie de blocs de longueur fixe, connue à la compilation $=>$ déroulement complet des boucles,
3. résultats directement dans la matrice ou en cache,
4. ordre des boucles...
5. e.t.c.

- Optimisation par tests systématiques, sans hypothèses autres que l'existence de caches L1, L2.

- Optimisation par tests systématiques, sans hypothèses autres que l'existence de caches L1, L2.
- batterie de tests successifs,
- résultats finaux figés pour obtenir la bibliothèque optimisée.

Performances

ARCH	ATLAS	COMP	% Peak	Peak Gflops
2.4Ghz Core2	3.7.18	gcc	78%	9.6
900Mhz Itanium2	3.6.0	icc	90%	3.6
1.6Ghz Opteron	3.6.0	gcc	88%	3.2
1062Mhz UltraSPARC III	3.7.8	gcc 3.3	82%	2.124
600Mhz Atdlon	3.5.7	gcc 2.95.3	80%	1.2
2.8Ghz Pentium4E	3.7.3	gcc 3.3.2	77%	5.6
2.6Ghz Pentium4	3.6.0	gcc	77%	5.2
1Ghz PentiumIII	3.7.7	gcc 2.95.3	76%	1
1Ghz Efficieon	3.7.7	gcc 3.2	60%	2
1.8Ghz PPC970FX (G5)	3.7.10	Apple gcc 3.3	69%	7.2
3.0Ghz P4E EM64T	3.7.10	gcc RH 3.2.3	78%	6.0

The Fastest Fourier Transform Of the West!

FFT classique: si $n = a^p b^q c^r ...$ alors le calcul se ramène à des FFT de tailles $a,\ b,\ c$.....

The Fastest Fourier Transform Of the West!

FFT classique: si $n = a^p b^q c^r \ldots$ alors le calcul se ramène à des FFT de tailles $a,\ b,\ c$.....
Ce calcul est optimal en terme de nombre d'opérations.

The Fastest Fourier Transform Of the West!

FFT classique: si $n = a^p b^q c^r \ldots$ alors le calcul se ramène à des FFT de tailles $a,\ b,\ c$.....
Ce calcul est optimal en terme de nombre d'opérations.
L'optimal absolu $n = 2^r$.
(Cooley-Tuckey 1965, après Gauss en 1805).
mais en pratique ce n'est **pas optimal!**

- accès non contigu à la mémoire non optimal.

- accès non contigu à la mémoire non optimal.
- implémentation naïve: récursivité; implémentation habituelle non récursive (cf. M. Schatzman “Analyse Numérique”).

- accès non contigu à la mémoire non optimal.
- implémentation naïve: récursivité; implémentation habituelle non récursive (cf. M. Schatzman “Analyse Numérique”).

Plans

Idées: précalculer une stratégie optimale:

- une factorisation de n (pas forcément en facteurs premiers),

Plans

Idées: précalculer une stratégie optimale:

- une factorisation de n (pas forcément en facteurs premiers),
- une stratégie mêlant récursion et approche non récursive,

Plans

Idées: précalculer une stratégie optimale:

- une factorisation de n (pas forcément en facteurs premiers),
- une stratégie mêlant récursion et approche non récursive,

Les résultats de cette stratégie sont sauvegardés (PLAN), et utilisés lors des calculs.

Plans

Idées: précalculer une stratégie optimale:

- une factorisation de n (pas forcément en facteurs premiers),
- une stratégie mêlant récursion et approche non récursive,

Les résultats de cette stratégie sont sauvegardés (PLAN), et utilisés lors des calculs.
Les opérations élémentaires sont effectuées avec des “codelets”, fragment de code optimisés (150).

Le “FFTW planner”

- le problème: trouver une factorisation optimale de n

Le “FFTW planner”

- le problème: trouver une factorisation optimale de n
- sachant que pour chaque facteur, plusieurs stratégies existent.

Le “FFTW planner”

- le problème: trouver une factorisation optimale de n
- sachant que pour chaque facteur, plusieurs stratégies existent.

Programmation dynamique:
trouver un “chemin” optimal dans les factorisations de n.

Performances

- petites et grands vecteurs: moins rapides..
- tailles 32–16000 (PIV 2.8 Ghz): 3 Gflops.