

OpenMP
Lionel Lacassagne

Introduction

Le but de ce TP est de mesurer l'impact de la parallélisation de code par OpenMP et de la comparer à la vectorisation. Les fichiers C nécessaires se trouvent à l'adresse www.ief.u-psud.fr/~lacas/Teaching.

1 Opérateurs 1D

1.1 Addition de deux vecteurs (tableaux 1D) de registres SIMD

Soient les vecteurs \vec{X}_1 et \vec{X}_2 de taille n et \vec{Y} leur somme:

$$\vec{Y} = \vec{X}_1 + \vec{X}_2 = \sum_{i=0}^{n-1} X_1(i) + X_2(i) \quad (1)$$

1. Coder cette addition en scalaire avec des `double`. dans la fonction `add_f32vector()`. La fonction de test (réalisant les allocations mémoire) est `test_add_f32vector()`
2. Activer la parallélisation de code par OpenMP, avec/sans vectorisation. Mesurer l'impact de chaque accélération.

1.2 Somme sur un voisinage de 3 points (tableaux 1D) de registres SIMD: réduction partielle

Les opérateurs précédents réalisent des calculs entre registres SIMD. Une difficulté du calcul SIMD est la réalisation de calcul à l'intérieur d'un registre SIMD.

Soit le tableau X . On souhaite que le tableau Y contienne en tout point la somme de trois points du tableau X :

$$Y_3(i) = X(i-1) + X(i) + X(i+1) \quad (2)$$

1. Codez cet opérateur dans la fonction `sum3_f32vector`. La fonction de test et d'allocation est `test_sum_f32vector`.
2. Activer la parallélisation de code par OpenMP, avec/sans vectorisation. Mesurer l'impact de chaque accélération.

2 calcul mathématique: calcul de π

Les techniques de calculs de π présentées ici sont volontairement inefficaces. Elles n'ont d'autre but que d'être pédagogiques. Elles nécessitent beaucoup d'itérations afin de mettre en évidence l'utilité d'OpenMP et de la vectorisation.

$$\frac{\pi}{4} = \arctan(1) = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots \quad (3)$$

$$\frac{\pi}{4} = \int_0^1 \frac{dx}{1+x^2} \Rightarrow \pi \simeq 4 \sum_{k=0}^{k=n} \frac{1}{1+(k/n)^2} \quad (4)$$

1. Coder les formules (3) et(4) dans les fonctions `pi_atan1` et `pi_integrale` en utilisant des `double`.
2. Mesurer le temps de calcul pour différentes valeur de n , le nombre d'itérations.
3. Ajouter des `pragma` OpenMP. et refaire les mesures de temps.
4. Modifier le Makefile pour activer la vectorisation de code.
5. Pour l'ensemble des versions scalaire, OMP, vec, vec+OMP, indiquer l'accélération. Conclusion.

3 Références sur le calcul de π par arc-tangentes

Il existent un grand nombre de formules basées sur le développement de l'arc-tangente(Eq. 5). La plus inefficace est $\pi/4 = \arctan(1)$ (Eq. 3) qui nécessite des milliards d'itérations pour seulement quelques chiffres.

$$\arctan(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{2k+1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots \quad (5)$$

Une formule plus efficace est celle de John Machin (1706) (Eq. 6)

$$\frac{\pi}{4} = 4 \arctan(1/5) - \arctan(1/239) \quad (6)$$

A titre de comparaison, sur une HP 49G+ (ARM7 à 48 MHz) disposant d'un noyau de calcul formel et en multi-précision, la formule de Machin permet de calculer 20 décimales en 1,8s et 50 décimales en 4,5s.

La meilleure formule connue à base d'arc-tangentes est donnée par l'équation 7: pour produire 1 chiffre supplémentaire, il faut *en moyenne* ajouter 1,58 termes. Pour comparaison, la formule de Machin nécessite d'ajouter 1,85 termes.

$$\frac{\pi}{4} = 44 \arctan(1/157) + 7 \arctan(1/239) - 12 \arctan(1/682) + 24 \arctan(1/12943) \quad (7)$$

Pour connaître les formules les plus efficaces (quadratiques ou d'ordre 3) : "Le fascinant nombre π , Éditions Belin, Pour la Science - (ISBN 2-902918-25-9)" de Jean-Paul Delahaye et les sites web <http://www.pi314.net/fr/index.php> et <http://bellard.org> (un des derniers recordmen du calcul de π).

4 Application numérique

$$\pi \simeq 3,141\,592\,653\,589\,793$$