Université de Paris Sud
———
Masters NSI & SETI

Architecture des Ordinateurs

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`.

Le fonctionnement des fichiers est le suivant: dans la fonction `main()`, il y a un appel à une fonction qui indique si le code est en mode `DEBUG` (pour la mise au point et les tests unitaire) ou en mode `BENCHMARK` (activation via `mymacro.h`) puis qui paramétrise le fonctionnement OpenMP, en forçant le nombre de threads créés à `p`. Il faut réaliser deux tests: l'un avec `p` égal au nombre de coeurs, l'autre en doublant la valeur de `p` pour tester l'impact de l'*HyperThreading*. Pour cela, il y a deux fichiers `Makefile`: l'un pour le monothreading, l'autre pour le multithreading préfixé par `_OMP`. Les fichiers `Makefile` doitvent être adaptés à l'architecture de la machine (option `-x`) en spécifiant `SSE2`, ... `SSE4.1`, `SSE4.1` ou éventuellement `AVX`. Attention, avec la version 9 du compilateur icc, limité en terme de jeu d'instruction SIMD reconnu, mettre `-msse2` à la place de `-xSSE4.2`, ce qui implique de codez en `SSE2`.

1 Ensemble fractal de Mandelbrot

Les buts de cet exercice sont de combiner SIMD et OpenMP pour obtenir le code le plus rapide possible. L'ensemble de Mendelbrot (du nom de Benoit Mandelbrot, Polytechnicien X-1944 et mathématicien chez IBM) est une fractale définie comme étant l'ensemble c du plan complexe tel que la suite par:

$$z_0 = 0 \quad (1)$$
$$z_{n+1} = z_n^2 + c \quad (2)$$

ne tend pas vers l'infini. Le module doit rester inférieur ou égal à deux. En remplaçant z_n par le couple de réels (x_n, y_n) et c par (a, b), nous obtenons:

$$(x_0, y_0) = (0, 0) \quad (3)$$
$$x_{n+1} = x_n^2 - y_n^2 + a \quad (4)$$
$$y_{n+1} = 2x_n y_n + b \quad (5)$$
$$|z_n| = \sqrt{x_n^2 + y_n^2} \quad (6)$$

Afin de donner une "couleur" aux points n'appartenant pas à l'ensemble (`http://en.wikipedia.org/wiki/Mandelbrot_set`), la méthode traditionnellement choisie est de mémoriser en tout point le numéro de l'itération associé à un module supérieur à 2. Afin de simplifier le benchmark, la fonction `main_mandelbrot` passe deux paramètres aux fonctions scalaires et SIMD: la taille n de l'image et le nombre max d'itérations à faire.

1. Codez en scalaire une fonction prenant un point (a, b) et un nombre max d'itération et renvoyant un entier qui vaut soit le nombre max d'itération, si la suite reste de module ≤ 2, soit le numéro de la dernière itération où le module est resté ≤ 2. Validez son fonctionnement via excel et les couples (a, b) de valeurs suivants: $(-1.00, +0.40)$, $(-0.90, 0.30)$ $(-0.80, 0.30)$ et $(-0.70, 0.10)$. le nom de la fonction sera `int mandelbrot_scalar(float a, float b, int max_iter)` et celui de la fonction de test `void test_mandelbrot_scalar(float a, float b, int max_iter)`.

2. Le benchmark de cette fonction est fait "àutomatiquement" (fonction déjà écrites) dans `bench_mandelbrot_scalar`. mesurez le temps obtenu pour différentes tailles d'images en modifiant le paramètre `size` (typiquement 512 ou 1024).

3. Afin d'accélérer la vitesse de calcul, ajoutez un pragma OpenMP dans la fonction `calc_mandelbrot_scal` initialement sans préciser d'ordonnancemment, puis en forçant un ordonnancement d'abord statique (`schedule(static)` puis dynamique (`schedule(dynamic)`, en précisant alors la taille du bloc. Récapitulez tous les résultats dans un tableau et analysez-les.

4. Faites de même en SIMD. Deux possibilités pour compter le nombre d'itération: soit en flottant (version la plus simple) soit en entier (version la plus rapide).

5. Faites les mêmes tests pour OpenMP. Récapituler tous les résultats dans un tableau et analyser-les.

6. Comparez finalement les différentes accélérations entre-elles: scalar+OpenMP vs SIMD+OpenMP. Conclure.

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.

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

$$\frac{\pi}{4} = \int_0^1 \frac{dx}{1+x^2} \Rightarrow \pi \simeq 4 \sum_{k=0}^{k=n} \frac{1/n}{1+(k/n)^2} \tag{8}$$

1. Codez les formules (7) et(8) dans les fonctions `pi_atan1` et `pi_integrale` en utilisant des `double`.

2. Mesurez le temps de calcul pour différentes valeur de n, le nombre d'itérations.

3. Ajoutez des `pragma` OpenMP. et refaire les mesures de temps.

4. Modifiez le Makefile pour activer la vectorisation de code.

5. Pour l'ensemble des versions scalaire, OMP, vec, vec+OMP, indiquez l'accélération. Conclure.

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. 9). La plus inefficace est $\pi/4 = arctan(1)$ (Eq. 7) qui nécessite des milliards d'itérations pour seulement quelques chiffres.

$$\arctan(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{2k+1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots \tag{9}$$

Une formule plus efficace est celle de John Machin (1706) (Eq. 10)

$$\frac{\pi}{4} = 4\arctan(1/5) - \arctan(1/239) \tag{10}$$

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 11: 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.

$$\frac{\pi}{4} = 44\arctan(1/157) + 7\arctan(1/239) - 12\arctan(1/682) + 24\arctan(1/12943) \tag{11}$$

Pour connaitre 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

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