Projet PMC
Stencil

Date de rendu : vendredi 6 avril 2018.

Objectif : Réaliser un calcul de stencil sur des grilles 2D et 3D en utilisant les unités vectorielles et plusieurs coeurs de calcul.

Le calcul de stencil, aussi appelé convolution, consiste à calculer la valeur d'un point en fonction de son voisinage. La forme du stencil définie les points du voisinage à considérer. Pour chaque voisin on applique un coefficient https://en.wikipedia.org/wiki/Stencil_code
En 2D, un stencil 5 points considère les 5 voisins directes d'un point auxquels on applique un coefficient pour calculer la nouvelle valeur en ce point.

Dimensions du problème : On considère une grille 2D de dimension 512x512 et une grille 3D de dimension 512x512x512 donc chaque point contient une valeur flottantes en simple précision. Les voisins à considérer sont les voisins immédiats sans les diagonales (5 points en 2D, 7 points en 3D). On effectue sur chaque grille 1000 itérations de calcul.

Réalisation des codes :
1. Réaliser les codes scalaires de stencil (2D et 3D) qui serviront à vérifier les résultats sur les codes optimisés.
2. Valider indépendamment les opérations vectorielles de décalage de registres sur un exemple simple.
3. Utiliser les opérations de décalage pour les intégrer dans le code de calcul. Valider le résultat avec le code scalaire.
4. Ajouter une directive OpenMP pour exploiter plusieurs coeurs

Mesure de performance et analyse des résultats :
1. Évaluer les performances de votre machine : bande passante mémoire à l'aide du benchmark Stream (https://www.cs.virginia.edu/stream/FTP/Code/). Compilation du benchmark : gcc -o stream stream.c -O3 -march=native -fopenmp. Pour tester la bande passante en fonction du nombre de coeurs faire varier le nombre de coeurs utilisés à l'aide de la variable d'environnement OMP_NUM_THREADS : export OMP_NUM_THREADS=2.
Pour les performaces du processeur, la puissance théorique pour le calcul flottant simple précision est évaluée à l'aide de la formule suivante : fréquence de base (Ghz) * nombre de coeurs * largeur unité vectorielle (8 pour AVX2) * 2 si FMA supportée (Fused Multiply Add) * nombre d'unités FMA (2 après génération Broadwell).
Pour voir les caractéristiques de votre processeur : ark.intel.com, mettre le modèle de votre CPU (obtenu avec lscpu ou cat /proc/cpuinfo) dans la barre de recherche.
2. Mesurer les temps d'exécution de vos codes, comparer les performances des versions scalaires, vectorielles, OpenMP, … Tester différents compilateurs avec différentes options de compilation (-O2, -O3, …).
3. Déterminer le nombre d'opérations arithmétiques nécessaires pour vos codes : pour la version scalaire, pour traiter une itération de stencil sur une grille de 512x512 il y a 510x510 points à traiter (sans les bords) et pour chaque point il faut 5 FMAs donc 10 opérations arithmétiques. Par contre il faut charger 512x512 nombres flottants soit 512x512x4 octets.
4. Comparer les valeurs obtenues pour déterminer si votre code est limité par la puissance de calcul ou par la bande passante mémoire.
5. Bonus : Tester et analyser d'autres optimisations (déroulage de boucles, stratégies OpenMP, …)

Rendu : codes (avec un Makefile ou un script pour compiler) + rapport contenant votre analyse des performances (graphiques + commentaires). Le tout à m'envoyer à sylvain.jubertie@univ-orleans.fr.