TP 2 : Évaluation de performance *(suite)* – Options de compilation – Vectorisation

Les objectifs de cette séance sont de réaliser des études de performance, et de se familiariser avec les options de compilation et la vectorisation. Les codes sont disponibles sur le site Internet du cours : https://perso.ensta-paris.fr/~modave/SIM203/

Exercice 1. Étude comparative de la performance arithmétique des routines BLAS

Ici, on fait une vraie étude de performance. ***Systématisez vos calculs pour gagner du temps !***

Prenez le code `dgemmMKL.c` ou `dgemmMKL.cpp` du premier TP.

1. Calculez systématiquement la performance arithmétique de la routine dgemm pour des matrices carrés de tailles différentes *($n \times n$, pour n entre 10 à 1000 – inutile de prendre toutes les valeurs).*
2. Effectuez une étude similaire pour la routine sgemm et pour les routines dgemv et sgemv *(Pour les deux dernières routines, qui sont des routines BLAS2, considérez une matrice carrée de taille $n \times n$, pour n entre 10 à 30000 – inutile de prendre toutes les valeurs).*
3. Comparez et interprétez les différentes courbes.

Exercice 2. Code de différences finies – Analyse du rapport de compilation

Les rapports de compilation donnent beaucoup d'information sur les parties de code qui auraient (ou pas) été optimisées par le compilateur. Cela permet de vérifier que des optimisations demandées ont été effectuées, ou bien d'avoir une explication sur une absence d'optimisation. Le programmeur peut alors modifier son code pour faciliter une optimisation.

Prenez le code `diffusionIcc.c` dont les résultats ont été montrés au cours.

1. Évaluer les temps de calcul pour les implémentations 1 à 3 en testant successivement les options de compilations `-O0`, `-O1`, `-O2` et `-O3`. Reproduisez le tableau du slide 13 du cours. Avez-vous des résultats similaires ?
2. Ensuite, étudiez les rapports de compilation pour chaque niveau de compilation. Essayez de comprendre le sens des différentes optimisations réalisées par le compilateur, et de les relier à la matière vue au cours. *(L'objectif n'est pas de comprendre tout, mais d'arriver à faire quelques liens.)*

Exercice 3. Code dgemm – Optimisation et analyse du rapport de compilation

À votre tour.

Reprenez la routine dgemm que vous avez écrite dans le cadre du premier TP.

1. Étudiez le rapport de compilation pour le niveau d'optimisation maximum, en particulier l'ordre des boucles et la vectorisation des instructions. Testez différents arrangements des boucles. Pouvez-vous interpréter les choix du compilateur ?
2. Calculez systématiquement la performance arithmétique de la routine pour des matrices carrés de tailles différentes *($n \times n$, pour n entre 10 à 1000 – inutile de prendre toutes les valeurs)*, et comparez avec les résultats de la routine dgemm de la librairie mkl. Interpretez les résultats.

Références

– Intel® C++ Compiler Classic Developer Guide and Reference – Optimization Report Options
https://software.intel.com/en-us/cpp-compiler-developer-guide-and-reference-optimization-report-options

– Intel® C++ Compiler Classic Developer Guide and Reference – Vectorization
https://software.intel.com/en-us/cpp-compiler-developer-guide-and-reference-vectorization