TP3

Objectif

Utilisation des « threads » pour optimiser les performances d'un programme de multiplication de matrices.

Travail à réaliser

1- Prendre une copie du programme mul_mat_th.c, et le compléter pour réaliser la multiplication des matrices MA et MB avec un ensemble de threads dont le nombre est défini par NB_TH.

Cette multiplication repose sur le découpage de la matrice MB en NB_TH blocs verticaux (découpage sur j), de manière à ce que chaque thread réalise la multiplication de la matrice MA par chacun de ces blocs de MB.

2- Compiler le programme avec : gcc -D_REENTRANT -o mmth mul_mat_th.c -lpthread

-D_REENTRANT pour que la fonction utilisée par les threads soit réentrante. *Une fonction **réentrante** peut être parcourue simultanément par plusieurs fils d'exécution sans que cela ne cause de problèmes. Elle ne conserve pas de données statiques entre appels successifs, ne retourne pas de données statiques, et n'appelle pas de fonctions non-**réentrantes***.

3- Exécuter le programme avec la commande /usr/bin/time -f "%E %P" ./mmth 2000

La commande time -f "%E %P" affiche le temps consommé par l'exécution de mmth, ainsi que le pourcentage d'occupation du processus (400% au maximum pour 4 coeurs).

Comparer le temps obtenu avec celui obtenu au TP2 (pour le même code).

Modifier le code afin d'assurer la meilleure utilisation du cache (dernière étape du TP2), compiler avec l'option –O3, exécuter et comparer avec les résultats du TP2.

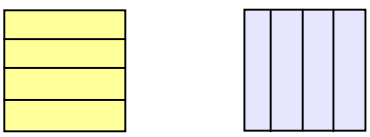
4- Pour gagner en performances, on peut travailler sur des copies des matrices dans la mémoire locale à chaque thread.

Dans la fonction associée aux threads, effectuer le calcul dans une sous-matrice C locale, puis copier le résultat final dans la matrice MC. Vérifier le temps d'exécution, et comparer avec les résultats précédents.

Serait-il utile de travailler sur des copies locales des matrices A et B ?

5- Refaire les étapes 3 et 4 en découpant chacune des matrices MA et MB selon les deux axes, respectivement I pour A, et J pour B.

Comparer les performances obtenues.



6- Comparer le dernier temps d'exécution avec celui de du premier algorithme du TP2 (sans aucune optimisation)