Jouons avec le cache

L'objectif de ce TP est d'observer quelques comportements dûs à la hiérarchie mémoire des machines contemporaines.

Vérifiez avec top qu'il ne traîne pas de processus « fou » sur votre machine qui perturberait les mesures. Éventuellement, redémarrez votre machine. Les tests doivent absolument durer plusieurs secondes et répétés plusieurs fois, car vos machines adaptent automatiquement la vitesse du processeur selon la charge de calcul demandée, les premières itérations sont donc lentes (il faut « chauffer » la machine).

Faites attention à ne même pas bouger la souris pendant la mesure : surtout dans un environnement KDE, toute action utilisateur entraîne une flopée de traitements :)

1 Accès mémoire

Lisez le programme `tab.c`. Son principe est d'allouer un tampon, d'aller taper dedans et de mesurer le temps d'accès à la mémoire. La variable *gold* est là juste pour taper *aléatoirement* dans le tampon sans pour autant que ce soit long à calculer (demandez Knuth pour les détails).

Tracez une courbe à l'échelle logarithmique (à l'aide de la commande `set logscale x 2`), à quoi correspond le saut de latence ? Raffinez vos mesures aux environ de ce saut pour déterminer assez précisement à combien il est.

2 Somme de matrices

Les effets de cache peuvent avoir des répercussions importantes sur l'exécution d'un programme. Pour cette section, écrivez un programme effectuant la somme de deux matrices. La façon classique de procéder consiste à parcourir les matrices de façon à minimiser le nombre de défauts de cache. Que ce passe-t-il si l'on inverse les deux boucles ? Essayez de quantifier le nombre de défauts de cache pour chaque cas.

3 Multiplication de matrices

Le programme `mul_mat.c` effectue une multiplication de matrice de façon très basique. Cette façon de faire est loin d'être optimale et les défauts de cache sont nombreux.

Écrivez une multiplication de matrice utilisant efficacement le cache du processeur. Le gain obtenu est-il décevant, correct ou plus que satisfaisant ?

4 Faux partage

Le programme `false_sharing écart coeur1 coeur2` lance deux threads sur les coeurs donnés en paramètre. Chaque thread exécutent une boucle qui décrémente une variable. L'écart en octets entre les deux variables décrémentées est précisé en paramètre. Le programme affiche le nombre de millions de décrémentations par seconde réalisés par chaque thread.

Utiliser ce programme pour mesurer les effets du faux partage en fonction de la position relative des différents threads.