OBHPC
TD2
Programmation C et mesures de performances

Pour une mesure de performance stable, il faut s'assurer que certaines contraintes sont respectées:

1 – S'assurer que le laptop est connecté au secteur
2 – S'assurer que le CPU tourne a une frequence stable (**cpupower**)
3 – Pinner le processus sur un coeur de calcul (**taskset** ou **numactl**)

cpupower: cette commande permet de fixer la fréquence (ou le gouverneur) d'un ou plusieurs coeurs de calcul du CPU.

taskset: cette commande permet d'affecter à un coeur de calcul, un processus ou thread en garantissant qu'il n'y aura pas de migration lors de l'exécution.

Exercices:

0 – Extraire les informations sur l'architecture cicble:
0.1 – **Information sur le CPU:**
 # lscpu
 # cat /proc/cpuinfo
0.2 – **Informations sur les caches de données:**
Il faudra consulter les fichiers dans les chemins suivant**:**
/sys/devices/system/cpu/cpu0/cache/index0/* pour le cache L1
/sys/devices/system/cpu/cpu0/cache/index2/* pour le cache L2
/sys/devices/system/cpu/cpu0/cache/index3/* pour le cache L3

1 – Lancer le programme **dgemm** et récolter les mesures de performance pour chaque version dans un fichier à part (1 fichier par version).

2 – Modifier le **Makefile** fournit afin de tester plusieurs flags d'optimisation les compilateurs **gcc, clang, icx,** et **icc** (si installés).

3 – Générer les fichiers de performance pour chacune des versions de la question 2.

4 – Rajouter une version de la fonction **dgemm** avec déroulage x8 et comparer ses performances aux autres versions.

5 – Générer des graphiques (histogrammes) avec **GNUPlot** comparant les différentes versions pour chaque compilateur et un graphique comparant les version par compilateur.

6 – Refaire les étapes 1 à 4 pour les codes **dotprod** et **reduc**.

7 – Fournir un rapport de 3 à 4 pages décrivant les résultats de vos expériences.