TP4 : produit matriciel et routines BLAS

Version du 11 janvier 2022

Récupérer le code du produit matriciel séquentiel $C+ = A \times B$ (les 3 matrices A, B et C sont toutes considérées comme étant carrées, d'ordre N). L'exécution pour différentes valeurs de N pourra se faire avec le script `run.sh`.

1. (**Produit matriciel naïf**) Sur votre machine, mesurez la performance (en Gflop/s) du produit matriciel "naïf" (3 boucles imbriquées), par exemple pour $N = 512$, et comparez la à la performance crête d'un cœur de votre machine.
   En plus des options `-march=native` et `-mtune=native`, déterminez parmi les options d'optimisation à la compilation suivantes de `gcc` celles qui permettent d'améliorer effectivement vos performances pour $N = 512$: `-O3`, `-ffast-math`, `-funroll-loops`, `-fmodulo-sched`.
   Une fois ces options d'optimisation déterminées, mesurez les performances de votre produit matriciel lorsque N varie à l'aide du script `run.sh`. Qu'observez-vous ?
2. (**Produit matriciel récursif par bloc**) Ecrire un pseudo-code C récursif implémentant un produit matriciel par bloc pour des matrices allouées dynamiquement en mémoire sous la forme de tableaux 1D. On supposera que N est une puissance de 2. La fonction récursive à implémenter aura la signature suivante :

```
mm (int crow, int ccol, /* Upper-left corner of C block */
    int arow, int acol, /* Upper-left corner of A block */
    int brow, int bcol, /* Upper-left corner of B block */
    int n,              /* Blocks are n x n */
    int stride,         /* Stride for whole matrix */
    float *A, float *B, float *C);
```

   Implémenter cet algorithme et déterminer les valeurs de `seuil` qui permettent d'obtenir de meilleures performances que le produit matriciel naïf vu ci-dessus.
3. (**Produit matriciel parallèle avec OpenMP**) Comment paralléliser le produit matriciel récursif par bloc avec OpenMP ? Comment les performances varient en fonction du nombre de threads utilisés ?
4. (**calcul SIMD**) La vectorisation automatique de `gcc` est activée avec `-O3`. En ajoutant l'option `-fopt-info-vec`, vérifiez si `gcc` a réussi à vectoriser une boucle de votre calcul.
   Obtenez-vous de meilleures performances en utilisant les directives de compilation d'OpenMP pour la vectorisation ?

   Facultatif : obtenez-vous de meilleures performances en utilisant les intrinsèques AVX2 ? (N'hésitez pas à discuter de votre stratégie de vectorisation avec votre enseignant. On pourra de plus se limiter à des valeurs de N bien choisies pour la vectorisation.)
5. (**BLAS**) A l'aide de la librairie OpenBLAS (installée sur les machines du Département Informatique, et installable au besoin depuis le site d'OpenBLAS), utiliser la routine BLAS adaptée pour effectuer ce produit matriciel.
   On pourra compiler directement ainsi :

```
$ gcc -Wall matmul_blas.c /usr/lib/x86_64-linux-gnu/libopenblas.so.0
```

   Quelle fraction de la performance crête d'un cœur de votre machine arrivez-vous ainsi à obtenir ? (veillez à bien positionner la variable d'environnement `OPENBLAS_NUM_THREADS` à 1)
   Toujours à l'aide de `OPENBLAS_NUM_THREADS`, quelle fraction de la performance crête totale de votre machine arrivez-vous ainsi à obtenir ?

   Documentation sur les BLAS :
   — BLAS reference card (Fortran) : `http://www.netlib.org/blas/blasqr.pdf`
   — Interface C des BLAS :
     — documentation : `http://www.netlib.org/blas/blast-forum/cinterface.pdf`
     — fichier en-tête : `cblas.h` disponible sur `http://www.netlib.org/blas/#_cblas` (déjà installé au Département Informatique)
   — OpenBLAS : `http://www.openblas.net/` et `https://github.com/xianyi/OpenBLAS`

