TP 2 - MN

OZENDA Thomas - FERREIRA Joffrey

Partie 1) Blas

Voici un tableau résumant tout ce que l'on a fait sur ce tp dans la partie blas, nous aurions aimé pouvoir faire plus mais les examens et les autres tp nous ont pris également beaucoup de temps. Nous avons alors fait certains choix quant aux choix des fonctions à implémenter, paralléliser et tester.
Nous avons mis en vert les cases des fonctions testés plus bas.

	sxxx	dxxx	cxxx	zxxx
swap	Parallélisé, vectorisé, testé	Parallélisé, vectorisé, testé	Parallélisé vectorisé, testé	Parallélisé, vectorisé, testé
copy	Parallélisé, testé	Parallélisé, testé	Parallélisé, testé	Parallélisé, vectorisé testé
dot	Parallélisé, vectorisé, testé	Parallélisé, vectorisé, testé	cdotu_sub : parallélisé, vectorisé, (vectorisé marche pas) cdotc_sub : parallélisé, vectorisé, (vectorisé marche pas)	zdotu_sub : parallélisé, vectorisé, (vectorisé marche pas) zdotc_sub : parallélisé, vectorisé, (vectorisé marche pas)
axpy	Parallélisé, vectorisé, testé	Parallélisé, vectorisé, testé	Parallélisé, vectorisé, testé	Parallélisé, testé, non vectorisé
gemv	Parallélisé, vectorisé, testé	Parallélisé, vectorisé (résultat faux), testé	pas implémenté	pas implémenté
gemm	Parallélisé, vectorisé, testé	Parallélisé, vectorisé, testé mais on a que des résultats faux	pas implémenté	pas implémenté

Analyse pour chaque fonctions :

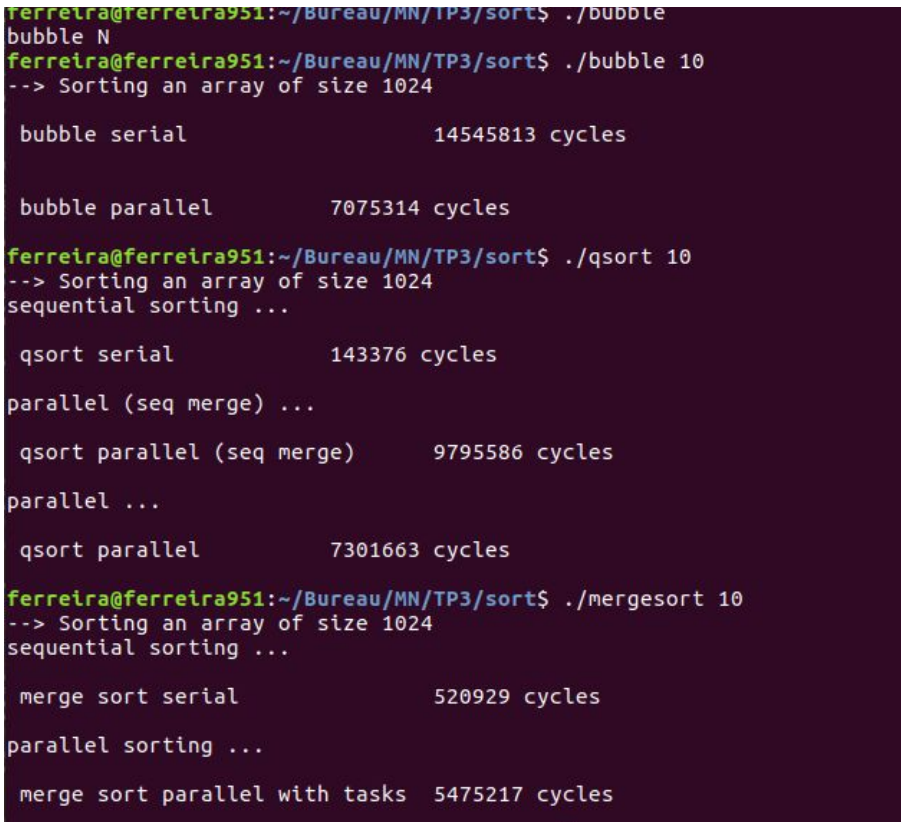
- dswap : On a des résultats semblables sauf pour la version séquentielle qui est moins performante.
- zcopy : Pour tous les tests de copy, on a des résultats bien meilleurs pour nos versions parallélisé et vectorisé. Même la version séquentielle est plus efficace que celle de cblas.
- cdotc_sub : La vectorisation permet de fortement améliorer les résultats de notre fonction, nos résultats sont équivalents pour cblas et mncblas_omp.
- caxpy : On a des résultats moins bon pour omp, même exécrable pour vectorisé car on passe par plusieurs vecteurs intermédiaires et effectue beaucoup d'opérations.
- sgemv : La fonction vectorisé est beaucoup plus efficace, la version omp a des bons résultats aussi, les deux autres fonctions ont les mêmes performances.
- dgemm : Les fonctions non parallélisés et vectorisé sont pas efficaces, la parallélisée a des résultats un peu moins bon que celle de cblas.

Partie 2) Tri

Explication de l'algo :

- bubble.c : Au début, nous avons fait une implémentation "simple" du tri à bulle (laissé en commentaire dans les fonctions) mais nous ne voyons pas comment la paralléliser. On a donc parallélisé sa variante le tri pair-impair : Il compte tous les couples d'éléments aux positions paires et impaires consécutives dans une liste et si un couple le premier élément est supérieur au second, il les échange de place. L'algorithme continue en alternant les comparaisons entre éléments aux positions paires-impaires et et aux positions impaires-paires consécutives jusqu'à ce que la liste soit ordonnée.
  On a parallélisé, les deux boucles for de notre programme avec la bibliothèque omp.
- qsort.c : on une fonction partition qui renvoie le pivot, ainsi qu'une fonction échanger qui permet de permuter 2 éléments d'un tableau. On appelle récursivement notre fonction de tri sur le sous tableau de l'élément 0 jusqu'au pivot, puis du pivot+1 jusqu'à la fin.
  On a seulement parallélisé ici le for dans la fonction partition.
- mergesort.c : On regarde si le tableau a qu'une valeur car ça veut dire qu'il est triée, sinon on le divise en deux tableaux quasi-égaux, ensuite on les tri récursivement et on les fusionne à la fin.
  On a parallélisé dans fusion le for qui permet de recopier les éléments du début du tableau.

Test des fonctions :



Analyse des résultats :

- Pour le tri à bulle, on divise par deux le nombre de cycles lorsqu'on parallélise notre fonction.
- Pour le tri rapide, on réduit le nombre de cycle d'environ 25% par rapport à la version séquentielle. Mais le qsort de la librairie C reste 5 fois plus rapide que notre version parallélisée.
- Pour le tri fusion, on a seulement parallélisé un for qui recopie un tableau dans un autre, mais la parallélisation a un impact négatif ça multiplie par 10 notre nombre de cycles.