8INF856
Programmation sur architectures parallèles
Devoir 2

(À remettre au plus tard mercredi le 30 octobre 2019)

- **Vous devez travailler en équipe de 2 ou 3.**
- **Vous serez évalué pour la rectitude et la qualité de vos réponses.**
- **Vous devez respecter scrupuleusement les instructions.**

1. Dans cet exercice vous devez implémenter l'algorithme récursif parallèle de fusion que nous avons vu en classe (PowerPoint "Introduction aux algorithmes parallèles", page 17) ainsi que l'algorithme séquentiel standard (powerpoint "Algorithmes diviser-pour-régner", page 4) . Vos programmes devront être écrits en C et vous devrez utiliser OpenMP. Sur l'entrée standard, votre programme doit lire un entier n ainsi que $2n$ entiers. Les n premiers entiers lus sont placés dans les n premières cases d'un tableau T et les n entiers lus suivants sont placés dans les n cases suivantes de T. Ce sont ces deux moitiés de tableaux que vous devez fusionner.

   Testez et analysez vos programmes. Le programme parallèle doit être testé et analysé avec 1, 2, 4, 8, 16, 24 et 48 threads. Dans chaque cas, indiquez à partir de quelle valeur de n votre solution parallèle est plus efficace que l'algorithme séquentiel (ne comptez pas le temps nécessaire pour initialiser T). Comparez aussi votre programme parallèle en fonction du nombre de threads utilisés. Quelle est le nombre de threads optimal? Expliquez vos résultats.

   Je vous demande de procéder de la façon suivante:

   - Dans votre répertoire maison sur `dim-openmpi0.uqac.ca` , vous devez créer un répertoire dont le nom est Devoir2

- Vos deux programmes devront être dans des fichiers d2s.c (séquentiel) et d2p.c (parallèle).
- Vos programmes doivent pouvoir être compilés sans faute à l'aide de la commande gcc. Pour votre programme parallèle utilisez la commande suivante: `gcc -fopenmp d2p.c`
- Vous devez me faire parvenir un document papier contenant les informations suivantes:
  (a) Le nom des coéquipiers
  (b) Le nom d'usager et le mot de passe du compte où se trouve le programme
  (c) Le résultat des tests ainsi que l'analyse.

Vos programmes seront testés à l'aide de la commande `a.out < fichier_test` où `a.out` est l'exécutable et `fichier_test` est un fichier contenant un entier n suivi de $2n$ entiers.

Remarque: Je vous suggère aussi d'utiliser la fonction `omp_get_wtime()` pour déterminer le temps d'exécution d'un thread. Il pourrait être aussi utile d'utiliser openMP avec votre programme séquentiel afin de chronométrer son temps d'exécution sur la même base que l'algorithme parallèle.

2. Implémentez l'algorithme parallèle du problème précédent en utilisant la librairie pthread au lieu de OpenMP. Placer votre programme dans le même répertoire que celui indiqué dans la question précédente. Le nom de votre programme doit être d2pthread.c. Votre programmes doit pouvoir être compilés sans faute à l'aide de la commande gcc. Pour votre programme parallèle utilisez la commande suivante:

```
gcc  d2pthread.c -lpthread
```