

# Algorithmique et Programmation Parallèle

## TD 1 OMP

Rappel :

- Pour compiler un programme C utilisant OpenMP avec le compilateur gcc :  

```
% gcc -fopenmp source.c -o monprog
```
- La variable d'environnement OMP\_NUM\_THREADS permet de spécifier le nombre de threads OpenMP :  
Possibilité 1 : 

```
% export OMP_NUM_THREADS=4 ; ./monprog
```

  
Possibilité 2 : 

```
% OMP_NUM_THREADS=4 ./monprog
```

### Exercice 1 : Passage de OpenMP à Pthread

1. Etudier, compiler et exécuter avec 4 threads le programme `omp_omp2pth/prog_omp.c`. Quelle doit être la valeur de la variable `sum` à la fin de l'exécution ?
2. Réécrire ce programme avec les threads POSIX.
3. Revenir sur le programme OpenMP. La boucle `for` est-elle parallélisée ? Modifier le programme pour que la boucle `for` soit distribuée sur tous les threads OpenMP. Quelle est la valeur attendue de la variable `sum` ?
4. OpenMP propose-t-il un autre moyen plus simple et plus efficace pour calculer `sum` ? Si oui, effectuer la modification.

## Exercice 2 : Produit matrice-creuse vecteur

Une matrice creuse est une matrice dont « la plupart » des éléments sont nuls. Dans notre cas, la matrice a au plus 5 éléments non nuls sur chaque ligne.

Pour compresser au maximum les données, on ne stocke que ces éléments (structure `sparse_matrix_t`). Pour une ligne `i`, on a :




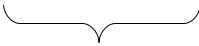
- le nombre d'éléments non nuls (i.e. le nombre de colonnes « utiles ») : `ncol[i]`
- le numéro des colonnes « utiles » : `col[i][k]` où  $0 \leq k < \text{ncol}[i]$
- les éléments non nuls : `elt[i][k]` où  $0 \leq k < \text{ncol}[i]$

On veut paralléliser le produit d'une matrice creuse par un vecteur. Pour mesurer le coût on effectue ce produit plusieurs fois :

`./monprog <N> <niter>`

où `N` est la dimension de la matrice  $N \times N$  (ou la taille du vecteur)

`niter` : nombre de répétitions du produits

0	1.2	0	0	0	0	1	1.2	1
0.3	0	0	-6.	0	0	2	0.3 -6.	0 3
0	0	2.7	0	0	0	1	2.7	2
0	-3.	0	0	9.1	5.2	3	-3. 9.1 5.2	1 4 5
0	0	0	4.	0	0	1	4.	3
0	0	-3.	0	0.8	0	2	-3. 0.8	2 4
								
Matrice						ncol	elt	col

1. Paralléliser la fonction `prod_mat_vec` en veillant à l'équilibrage des charges.
2. Paralléliser la fonction `is_equal`.
3. Relever les temps elapsed pour 1, 2, 4 et 8 threads OpenMP pour les paramètres suivants : `N = 1000000` et `niter = 100`. Commenter.