TPs 1&2

Master 2 SID
Benoist GASTON
benoist.gaston@gmail.com

OpenMP Multiplication Matricielle

- Considérer le programme **`prodmat`** (https://github.com/benoistgaston/openmp.git) qui effectue une multiplication de deux matrices A et B en stockant le résultat dans une matrice C. Il est composé de plusieurs séquences de calcul sous forme de boucles sur les indices des matrices.

- On se propose de partager les calculs entre différents threads OpenMP.

- **Questions**

  1. Prendre en main le code ; le compiler à l'aide du makefile.

  2. Identifier les boucles à paralléliser et positionner les directives OpenMP **`parallel`** et **`for`** (en utilisant un schedule **`runtime`**)

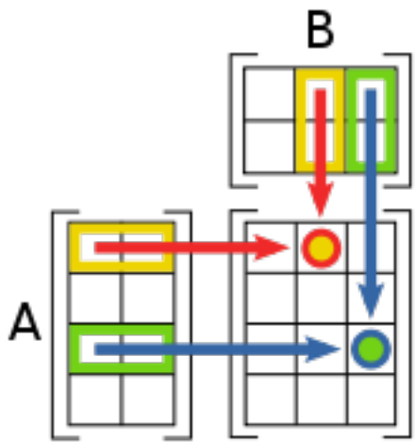
  3. Modifier le makefile afin d'intégrer l'option openMP

  4. Compiler et exécuter en jouant à l'aide de variable d'environnement sur le nombre de threads et sur le schedule

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} B = \begin{pmatrix} b_{11} & \cdots & b_{1p} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{np} \end{pmatrix}$$

$$AB = C = (c_{ij})_{n \times p}$$

$$c_{ij} = \sum_{k=0}^{n} a_{ik} \times b_{kj}$$

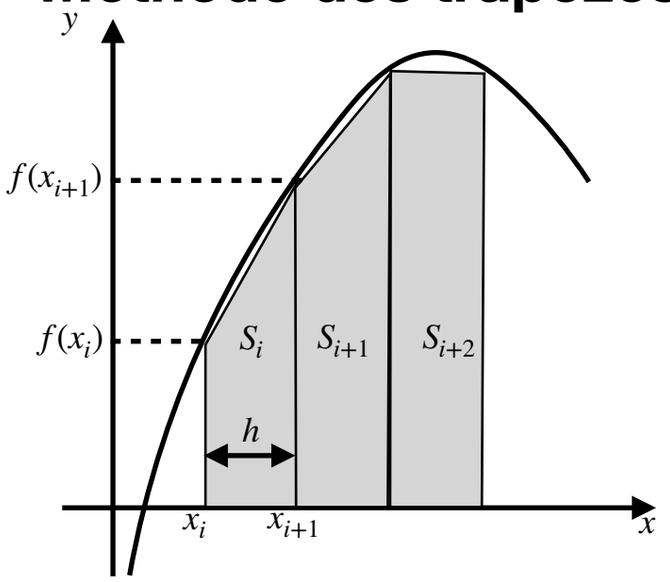


OpenMP Calcul intégral

- Considérer le programme **`integcos`** (https://github.com/benoistgaston/openmp.git) qui effectue le calcul de l'intégrale la fonction $\cos^2$ sur l'intervalle $[0,\ldots,\pi/4]$ par la méthode des trapèzes.
- Rappel : la valeur de cette intégrale est égale à $\pi/8 + 1/4$
- On se propose de partager ce calcul entre différents threads OpenMP.
- **Questions**

  1. Prendre en main le code ; le compiler à l'aide du makefile.
  2. Insérer les directives OpenMP appropriées dans le fichier **`integcos.c`**. La zone parallèle est déjà définie, il reste à insérer les directives de partage des données et du travail. On utilisera les directives : **`section`**, **`single`**, **`for`** et **`reduction`**.
  3. Analyser les performances de la version parallèle.

Méthode des trapèzes



Formule pour cos²

$$\int_0^{\frac{\pi}{4}} \cos^2(x)dx = \frac{1}{2}\cos^2(0) + \cos^2(h) + \cos^2(2h) + \cdots$$
$$+\cdots + \cos^2((n-1)h) + \frac{1}{2}\cos^2(nh)$$

OpenMP Fibonacci

- Considérer le programme **`fib.c`** (https://github.com/benoistgaston/openmp.git) qui calcul de manière récursive la suite de fibonacci.

$$F_n = n, n = 0,1$$
$$F_n = F_{n-1} + F_{n-2}, n \geq 2$$

- On se propose de paralléliser la fonction **`fib_rec()`** avec openmp sur le paradigme de distribution de tâches. Ce paradigme reprend l'exemple divide and conquer présenté en cours.

- **Questions**

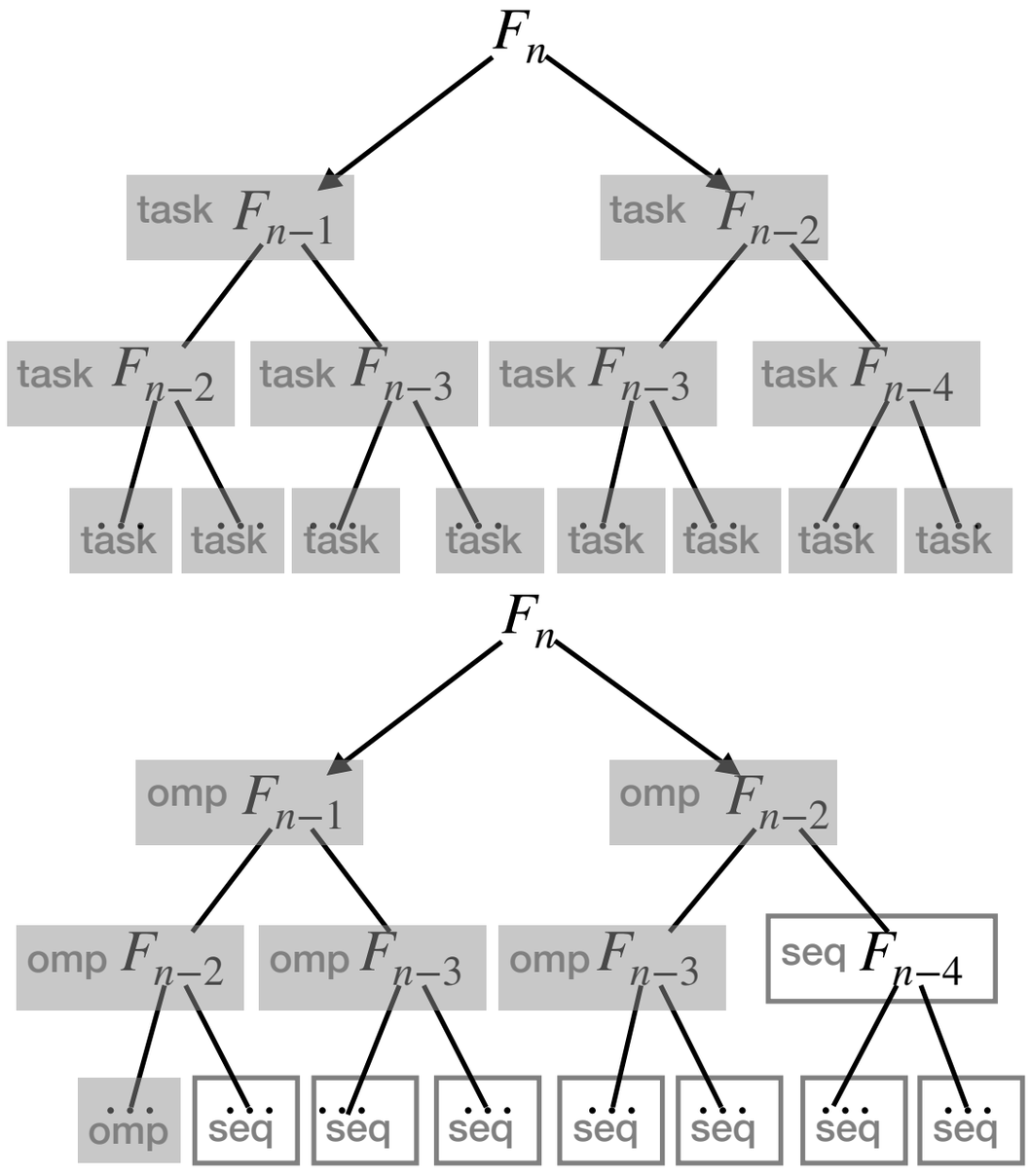
1. Prendre en main le code ; le compiler à l'aide du makefile. Faire tourner pour des valeurs de n 10, 20, 30 40.

2. Sur la base de la fonction **`fib_rec()`**, écrire une fonction **`fib_omp()`** parallélisant à l'aide de tâche.

3. Observer les performances de la version parallèle.

4. Pour résoudre le problème de performance constaté, définir dans **`fib_omp()`** un seuil en dessous duquel la fonction **`fib_rec()`** sera appelée à la place de **`fib_omp()`**.

5. Observer les performances de cette version hybride.



OpenMP Bubble Sort

- Considérer le programme **`bsort.c`** (https://github.com/benoistgaston/openmp.git) qui propose une implémentation du tri à bulle.

```
tri_à_bulles(Tableau T)
   pour i allant 1 de (taille de T)-1
       pour j allant de 0 à i-1
           si T[j+1] < T[j]
               échanger(T[j+1], T[j])
```

- On se propose de paralléliser la fonction **`bsort()`** avec openmp.
- **Questions**

1. Prendre en main le code ; le compiler à l'aide du makefile.
2. Tenter une directive parallel for sur la boucle interne.
3. Y a-t-il une erreur à la compilation ? À l'exécution (tester plusieurs fois) ?
4. Comment peut-on modifier l'algorithme pour la boucle for parallélisable ?
5. Modifier l'algorithme et le paralléliser.