Lab 2
OpenMP

Master SID 2 SD
Benoist GASTON
benoist.gaston@univ-rouen.fr

OpenMP

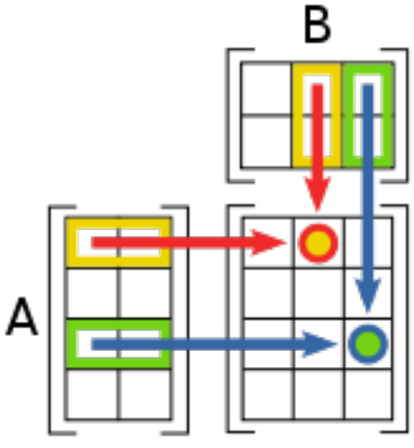
Matrix Multiplication

- The program `prodmat.c` multiples two matrices A and B and store the results in a matrix C. It is composed of several nested loops on the rows and columns of the matrices.
- We want to share the computing loop between different OpenMP threads.
- Questions
    1. Get started with the code; compile it with the makefile (command `make`).
    2. Identify loops to parallelize and put OpenMP directives `parallel` and `for` (choose `runtime` schedule)
    3. Compile the new source code with the makefile and run the openMP version. Modify environment variables to change number of threads and scheduling type.

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

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

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



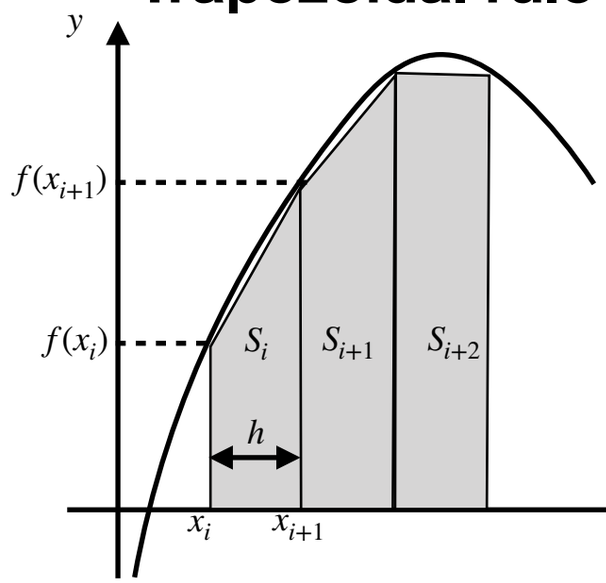
OpenMP

Integral

- The source code `integcos.c` do the numerical integration of function $\cos^2$ on the interval $[0,\ldots,\pi/4]$ using trapezoidal rule.
- Reminder : the value of this intégral is equal to $\pi/8 + 1/4$
- We want to share the computing loop between different OpenMP threads.
- Questions

    1. Get started with the code; compile it with the makefile (command `make`).
    2. Insert appropriate OpenMP directives in the source code `integcos.c`. The parallel region is allready define, just insert directives to distribute work and share data. Constraint : use the following directives : `section`, `single`, `for` and `reduction`.
    3. Analyse the performance of the parallel version.

Trapezoidal rule



Formula for cos²

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

OpenMP

Recursive Fibonacci

- The source code `fib.c` compute the *nth* value of Fibonacci applying the recursive algorithm: $F_n = n, n = 0,1$
  $F_n = F_{n-1} + F_{n-2}, n \geq 2$
- We want to parallelize the function `fib_rec()` with openmp using task parallelism. For that we will use the *divide and conquer* paradigm.
- Questions
  1. Get started with the code; compile it with the makefile (command `make`). Run it for different values of n: 10, 20, 30 and 40.
  2. Based on the function `fib_rec()`, implement a function `fib_omp()` using the openmp directive `task`.
  3. Observe the parallel version performances (using command `time`).
  4. To solve the performance issue, define in `fib_omp()` a threshold below which the non parallel function `fib_rec()` is call instead `fib_omp()`.
  5. Observe performance of this hybrid version.