



# OpenMP Exercises

**Massimiliano Culpo** - [m.culpo@cineca.it](mailto:m.culpo@cineca.it)

**CINECA - SuperComputing Applications and Innovation Department**



# Warm-up with OpenMP

## Basic skills

- ① Write a serial "Hello world!"
- ② Add OpenMP directives to have each thread prompt his greeting
- ③ Add a conditionally compiled header to show if OpenMP was enabled
- ④ Experiment with the `OMP_NUM_THREADS` environment variable

## Loop and loop scheduling

- ① Write a program to replicate the scheduling plot seen in the lecture
- ② Construct a  $n_{\text{threads}} \times n_{\text{iterations}}$  matrix to log who executed what
- ③ Write the information to the ASCII file `IterationMap.txt`
- ④ Use the script `draw.sh` to plot your results

# The everyday duty

## Code parallelization

- 1 Parallelize the serial code `pi.c` that computes the value of  $\pi$
- 2 Parallelize the serial code `laplace.c` that solves a 2D Laplace equation
  - start with an incremental approach
  - try to include the while loop inside the parallel region

## Bug busting

- 1 Find and correct the bugs in the sample programs
- 2 Try to explain what was causing the incorrect behavior

# The insane teaser

## Who am I (without library calls)?

- 1 Write an implementation for the two prototype functions:
  - `int get_num_threads()`
  - `int get_thread_id()`
- 2 You **can't use** library calls or explicit locks
- 3 The implementation must work for **nested parallel** regions
- 4 You can use all the directives you want
- 5 Thread ID must be consistent with the OpenMP library

## Hints

- 1 Write first an implementation that works for a single level of parallelism
- 2 Exploit data sharing attributes to exchange information between threads
- 3 Remember where barriers are implied