Lab 07 - Parallel programming using OpenMP (1)

(in C/C++)

0. Introduction

We will use the C/C++ programming language with OpenMP support.
So parallel tasks are now parallel threads (tasks are threads in this scenario).
Tutorials:
https://en.wikipedia.org/wiki/OpenMP
https://computing.llnl.gov/tutorials/openMP/

1. Programming Setup

Use whatever OS and C/C++ compiler with OpenMP support.

2. PROBLEM 1

Consider problem 4 from Lab01.pdf – computing PI using a Monte Carlo approach; the problem is nearly embarrassingly parallel.

Let's consider the total number of points to be very large, say X = 256*UINT_MAX
(quick question: what type must X have ?)
Given N threads, each thread will compute X / N iterations in a "parallel for".
You can play with the schedule of the "for" directive: static, dynamic, guided.
Select the one that gives the best results for your configuration.
Hint: use the "reduction" clause, too!

Like before, record the execution time in the following table and compute the relative speedup and relative efficiency (formulas are given in the table below, M is the number of equivalent logical cores):

THREADS	Execution time [milliseconds]	[Relative] Speedup S(n) = T(1)/T(n)	[Relative] Efficiency E(n) = S(n) / M
1		----------	----------
2			
4			
8			
16			

TABLE 1. Performance parameters for Problem 1

3. DELIVERABLES

3.1. A Word (or PDF) document containing TABLE 1; please specify clearly the number of cores of the computer you used for testing; also, please verify whether it has hyperthreading or not. Compute M, the number of "equivalent cores" as we did for the previous lab (as 1.3* nr of physical cores).
3.2. The C/C++ source code as a separate file.