ECE445: PARALLEL AND DISTRIBUTED COMPUTING
Fall 2021-2022

Homework 2:
OpenMP Programming
(*Deadline: Friday 14.01.2022*, 23:59)

General Instructions:
Follow the template text (template.docx) since this is part of your grade.
Write clearly your names and your IDs on the first page.
In the theoretical exercises, write in detail your answers and the sources from which you drew them.
In the programming ones, present and explain the algorithm you programmed and present in detail the experimental results of your programs.
Describe your implementation and the machine you worked on (hardware features, OS, compiler, etc.).
Program in C and Linux, use Makefile to compile / link / execute your programs.
Report your observations and present the results with tables and graphs.

Graphs and figures must be software-generated and not handmade and scanned.

In general, your text should be well written and legible, while you should FULLY justify the steps you followed, and comment on the results of each exercise. (20 credits)

Submit your work via eclass to a zip file (hw2_id1_id2_id3.zip), which will contain the text with your solutions / results and comments, your code (.c, .h files and your Makefile).

It is noted that the teams do not change during the semester.

Question 1.
Write a program in C (ask1.c) that implements parallel programming with OpenMP and calling the appropriate functions will display on the screen:
a) (5) how many processors the computer running the program has
b) (5) maximum number of computer threads
c) (5) number of threads participating in the execution
d) (5) message from the master thread: «Hello. I am the master thread. »
e) (5) message from the other threads: «Hi there! I am thread * », where * = their id
f) (5) the time of execution.
g) (10) Add a table inside your report, which will show the number of threads and the corresponding execution time on your computer.

Hint: Avoid id-based controls and use the appropriate functions to vary execution by thread.

Exercise 2.
Write a program in C (ask2.c) that implements parallel programming with OpenMP and calculates an approximation of π, using the formula

$$\pi = \int_0^1 \frac{4}{1+x^2}\,dx$$

Use Simpson's rule to approach the integral, i.e.,
if $x_i = i*dx$, i=0,1,…,N, dx= 1/N, a partition of [0,1] with N intervals (N + 1 points)

$$\pi = \int_0^1 \frac{4}{1+x^2}\,dx \approx \frac{dx}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + \cdots + 2y_{N-2} + 4y_{N-1} + y_N)$$

where $y_i = \frac{4}{1+x_i^2}$, $\mathrm{i} = 0,1,2,\ldots,\mathrm{N}$.

Your program should:

a) (10) run repeatedly in terms of the number of threads, from 1, 2, 4, 8, 16 and 32. In each iteration:
b) (10) The sum will be broken into a partial sums and each thread will calculate a partial sum. Try chunksize = 10, 100, 1000.
c) (20) The assignment of the terms of the sum to the threads will be static / dynamic (schedule=static/dynamic). Avoid checks based on which repetition is performed to have the appropriate factor in the sum as this will cause unnecessary delays.
d) (10) Measure the execution time in each repetition and the number of operations per sec for all chunksizes and scheduling type. The message should appear on the screen::
   pi is approximately * , computations time = *, number of threads = *, FLOPS = *, chunk = * , scheduling = *
e) (10) Add a table inside your report, which will show the number of threads and the corresponding execution time on your computer for all chunksizes and scheduling type. Comment on the results. Are the times similar or different and why?

Hint: FLOPS = number of operations per sec. Use N = 10^8.

Exercise 3.

Write a program in C (ask3.c) using OpenMP, which implements the Jacobi method for system solution

$$Ax = b \Leftrightarrow \begin{bmatrix} 2 & -1 & \\ -1 & \ddots & -1 \\ & -1 & 2 \end{bmatrix} x = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ N+1 \end{bmatrix}$$

where N is the dimension of the system.

Read slides 8-11 of 02_parallel_algorithms.pdf on the course website (eclass → documents) for more information on the Jacobi method.

Your program should:

a) (20) runs for M iterations
b) (40) in each iteration it should calculate the new approach of the solution, xnew, (and display on the screen) the remainder of the approach, i.e. $|| \text{b - A xnew} ||_2$, and the difference of the last 2 approaches, i.e. $|| \text{xold - xnew} ||_2$, with appropriate comments, e.g.:
   iter = *, residual = *, difference = *
c) (20) also display the execution time.
d) (20) Run your program for different number of threads (1, 2, 4, 8, 16, 32) and add a table to your report, which will show the number of threads and the corresponding execution time on your computer.

HInt: The system solution is the vector [1 2 3… N]. Use N = 10^6 and M = 20