Parallel Programming

Part A (30%)

Use opencv to write a C++ program named simulation.cpp to simulate the temperature changes according to the static heat distribution in a two-dimensional space. Consider a classroom with a rectangle shape. The walls have a fixed temperature +20C. There is a heater along one side of the wall with a fixed temperature +100C. The classroom is initially cold with -20C.

For any location that does not have a fixed temperature, its temperature in the next moment could be calculated by taking the average of the current temperatures of its four neighboring locations in vertical and horizontal directions. You can use either one 2D array (Gauss-Seidel iteration) or two 2D arrays (Jacobi iteration) to do the temperature update in each iteration.

The simulation terminates when the maximum difference between iterations at any location is less than a predefined threshold *epsilon*.

Use colors to represent different temperatures for the simulation. Use pure green to denote the lowest temperature you consider, and pure red the highest. Pure white would refer to a temperature in the middle.

The width and height of the rectangle, representing the size of the classroom, should be command line arguments.

Part B (70%)

Modify your simulation.cpp program into sequential.c. This time, no simulation is needed. Your program should have terminal output of (i) the total number of iterations; and (ii) the total computation time in ms. Your program should also write to a file named mat-sequential.dat, the final temperatures in each location. The output temperature should be organized one row per line, with tabs in each line to separate the different temperatures in a row.

Modify your sequential.c into parallel.cu. In addition to the width and the height of the rectangle, your program should have three other command line arguments: (i) the number of iterations to be performed; (ii) the number of blocks; (iii) the number of threads per block. It should print the computation time on the terminal and write the final temperatures into mat-parallel.dat following the same format of mat-sequential.dat. The program terminates after the given number of iterations. This number is obtained from the execution of sequential.c. Note that barrier synchronization is needed between iterations.

Use Bash command *diff* to check your output temperatures from sequential and from parallel executions.

Run your sequential.c and parallel.cu with the same size of the classroom to compare the efficiency. List in a table the comparison of the efficiency with at least five difference classroom sizes and/or different kernel launch configurations. The configuration refers to the number of blocks and the number of threads per block.

Submission: upload one zipped file called YourLastName_YourFirstName containing four plain text files:

- simulation.cpp
- sequential.c
- parallel.cu
- readme.txt

Your readme file should contain the comparison table and all explanations needed to compile, to execute, and to understand your program. There will be -10% penalty if the GA needs to contact you individually for more information in this regard.