CENG342 Project-2

Due: April 24, 2023

Penalty for late submission is 20% per day.

You cannot share or upload any of the assignments of CENG342 to any web page, or forum site or send somebody who does homework for money. If you take such an attitude, you will be in opposition to the law on the protection of personal data and violation of copyrighted verbatim. In addition, disciplinary action will be arranged by the Rules and Regulations Governing Student Disciplinary Actions in Institutions of Higher Education.

This is a teamwork assignment; you are going to work in groups of 3-4 students.

In this project, you are going to implement a parallel image processing algorithm. You are going to utilize multi-process parallelism via the OpenMP library. Image processing tools provide an extensive range of different algorithms for all kinds of circumstances. A simple and effective algorithm is **resizing an image**. It changes the size of the input image and produces another image in the desired size.

For the sake of simplicity, you only implement a parallel program that downscales an image to half of its size. Before writing a parallel program, you must first write a sequential program. Then you should make it parallel.

The program takes three inputs: the first for the input image, the second for the output image, and the third for the number of threads. Your program should perform a parallel **downscaling** algorithm on the input image and then store the results in a new image.

You need to measure the time for only **downscaling** and make tables that compare the timings of your algorithms with varying numbers of threads and different scheduling methods. In addition, you must make 2 more tables for speed-up and efficiency values for each experiment.

There are many downsizing algorithms already proposed in the literature, such as bilinear, bicubic, sinc, Lanczos, and so forth (Do not use Nearest Neighbor and sampling). You are free to choose any algorithm in your program.

You may look at this link:
https://towardsdatascience.com/image-processing-image-scaling-algorithms-ae29aaa6b36c

*If the size of an input image is 400x400 then the size of the output image should be 200x200.

What is required?

1. Report
   - A report that includes your names and surnames and appropriate title and small description.
   - Very short pseudocode of your **parallel** algorithm with at most **12 lines**.
   - Brief explanation: Steps of your parallel algorithm in terms of 4 steps of Foster's methodology.
     - Furthermore, you should discuss which parallelism you adopted, task or data parallelism, with the reasoning behind it.
   - Tables for experiments on your PC; for elapsed timings, speed-ups, and efficiencies of your parallel algorithm. For accurate timing please **take an average of at least 3 tests.**
   - Other tables for experiments on the server (16-core) that is provided for you to run your parallel code on it.
2. Source codes named
   a. seq_main.c,
   b. openmp_main.c
   c. hybrid_main.c

3. All source codes must be written in C/C++ programming language for sequential and parallel programs, respectively. Provide a Makefile description for compiling the source codes.

4. You must test your OpenMP program with the following `for` schedules:

- Default
- (Static,x): test chunk sizes for 1 and 100
- (Dynamic,x) test chunk sizes for 1 and 100
- (Guided,x) test chunk sizes for 100 and 1000

5. Draw speedup charts for every schedule. In your report, there must be 7 speed-up charts; one for each of the schedules. In the conclusion part, you must specify which schedule gives the best time according to your experiments

6. In Hyrid parallelism, at the outer level, you may use MPI parallelism. In each MPI process, you may use OpenMP parallelism. The total number of cores that is used in the experiments should not exceed the available cores in the system. Therefore, if there are 16 cores, some of the possible runs are:
   a. **mpirun -n 16 ./hybrid papagan.jpg output.jpg 1**
   b. **mpirun -n 8 ./hybrid papagan.jpg output.jpg 2**
   c. **mpirun -n 4 ./hybrid papagan.jpg output.jpg 4**
   d. **mpirun -n 2 ./hybrid papagan.jpg output.jpg 8**
   e. **mpirun -n 1 ./hybrid papagan.jpg output.jpg 16**

7. Required (20 pts): The CPU installed in the server is heterogenous that is 8 cores of 16 cores is faster than the other 8 cores (8 Performance and 8 Efficiency cores). Find a way to efficiently distribute your work on the cores so that load-balance on the cores will be optimum. Thus, you can reach the maximum speed-up.

Notes:

- Working together with other groups is prohibited.
- Submit your codes and your report (**PDF format**) in a zip file named project2-GroupX.zip into AYBUZEM. (Replace X with your group number.)
- One member in each group should upload the exam on behalf of their group. That is, there should be only one submission for each group in the Aybuzem.