Performance oriented software project

J. Daniel García Sánchez (coordinator)

Computer Architecture
ARCOS Group
Computer Science and Engineering Department
University Carlos III of Madrid

1 Objective

This project has as a fundamental goal that students get awareness with **sequential program optimization** as well as with **parallel programming models** in shared memory architectures.

Specifically, the project will focus in the developing optimized sequential software in the C++ programming language (including up to C++17), as well as in the use of parallel programming techniques using *OpenMP*.

2 Project description

In this project you will develop an image filtering application. You will develop two versions: a sequential version and a parallel version.

In the following sections, you will find the requirements that need to be satisfied. Additionally, a binary file with a version of an executable program with the required functionality will be published. This will make easier to compare expected results.

2.1 General overview

The application applies two filters on a set of images in BMP format. The result will be a new image with the same format, after applying the mentioned filters.

For every image the following filters will be applied:

- Gaussian blur.
- Sobel operator.

Two different versions of the program will be developed:

- **image-seq**: Sequential version.
- **image-par**: Parallel version.

Application (**image-seq** or **image-par**) will take 3 parameters:

- Execution mode.

- Directory with input images.
- Directory were all output images will be written.

Execution mode will be one of the following:

- **copy**: No transformation is performed. It only copies files.
- **gauss**: Applies the Gaussian blur filter.
- **sobel**: Applies the Gaussian blur filter and then the Sobel operator.

In any case, the application reads all files in the input director, applies the corresponding transformations and writes the corresponding files in the output directory.

The program shall check that the number of parameters is 3:

```
$image-seq
Wrong format:
  image-seq operation in_path out_path
    operation: copy, gauss, sobel
$
```

If the first argument does not take an adequate value (**copy**, **gauss** o **sobel**), An error message will be output and the program shall terminate:

```
$image-seq gaus indir outdir
Unexpected operation:gaus
  image-seq operation in_path out_path
    operation: copy, gauss, sobel
$
```

If the input directory does not exist or it cannot be opened, the program shall output an error message and it shall terminate:

```
$image-seq sobel indir outdir
Input path: inx
Output path: out
Cannot open directory [inx]
  image-seq operation in_path out_path
    operation: copy, gauss, sobel
$
```

If the output directory does not exist, the program shall output a message error and it shall terminate:

```
$image-seq sobel indir outx
Input path: indir
Output path: outx
Output directory [outx] does not exist
  image-seq operation in_path out_path
    operation: copy, gauss, sobel
$
```

In any other case, all files in the input directory will be processed and results will be written to the output directory. For each processed file the program shall print the timing information for every task. All values will be expressed in microseconds:

```
$image-seq sobel indir outdir
Input path: indir
Output path: outdir
File: "indir/62096.bmp"(time: 73016)
  Load time: 3442
  Gauss time: 38909
```

```
  Sobel time: 29022
  Store time: 1642
File: "indir/169012.bmp"(time: 65724)
  Load time: 2602
  Gauss time: 32950
  Sobel time: 28515
  Store time: 1655
$
```

2.2 BMP format

The BMP image format will be used. Files in this format contain a header of at least 54 bytes with the information specified in table 1.

Start	Length	Description
0	2	Characters **'B'** and **'M'**.
2	4	Size of file.
6	4	Reserved.
10	4	Start of image data.
14	4	Size of the bitmap header.
18	4	Width in pixels.
22	4	Height in pixels.
26	2	Number of planes.
28	2	Point size in bits.
30	4	Compression.
34	4	Image size.
38	4	Horizontal resolution.
42	4	Vertical resolution.
46	4	Color table size.
50	4	Important color counter.

Table 1: Header fields for BMP format.

The following constraints shall be met to consider an image as valid:

- The number of planes must be **1**.
- The point size must be **24** bits.
- The compression value must be **0**.

Note that there are fields that will not be used.

Pixels are stored starting in the location specified in the header field *start of image data*. Each pixel is represented by 3 consecutive bytes representing the blue, green and red value for that pixel. Those values will always be in the range **0** to **255**.

When pixels in a row are finished, a padding will be left before starting the next row. The padding will be such that the next row starts at a word boundary.

2.3 Gaussian blur

Gaussian blur reduces image noise and also image detail. It is usually used as a previous step to edge detection.

In general, a square matrix is used as a mask. For a mask of 5×5 the resulting image res, is obtained from the mask m and the original image im applying the following expression.

$$res(i,j) = \frac{1}{w} \sum_{s=-2}^{2} \sum_{t=-2}^{2} m(s+3, t+3) \cdot im(i+s, j+t)$$

where the mask matrix m and the weight w are:

$$m = \begin{pmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{pmatrix} \qquad w = 273$$

Important: Note that when applying the mas in image boundaries, the image will be considered as it was surrounded by imaginary pixels with value **0**.

2.4 Sobel operator

Sobel operator is an edge detection filter.

A mask of size 3×3 will be used.

Mask matrices m_x and m_y shall have values:

$$m_x = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \qquad m_y = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

Partial results res_x and res_y may be expressed as:

$$res_x(i,j) = \frac{1}{w} \sum_{s=-1}^{1} \sum_{t=-1}^{1} m_x(s+2, t+2) \cdot im(i+s, j+t)$$

and

$$res_y(i,j) = \frac{1}{w} \sum_{s=-1}^{1} \sum_{t=-1}^{1} m_y(s+2, t+2) \cdot im(i+s, j+t)$$

where

$$w = 8$$

Finally, result image is obtained as:

$$res(i,j) = |res_x(i,j)| + |res_y(i,j)|$$

3 Tasks

3.1 Sequential version

In this task you will develop a sequential version of the described application in C++17.

All your files must compile without problems and shall not emit any warning from the compiler. Specifically, the following compiler flags shall be enabled:

- `-std=c++17 -Wall -Wextra -Wno-deprecated -Werror -pedantic -pedantic-errors`

Please, also note that all evaluations shall be performed with compiler optimizations enabled (compiler options `-O3` and `-DNDEBUG`), as well as disabled (compiler option `-O0`).

Important: You are not allowed to use any library external to the C++ standard library.

3.2 Sequential performance evaluation

In this task you will carry out a performance evaluation for the application with both compiler optimizations enabled and disabled.

To evaluate performance you will have to measure the execution time for the application. You will represent graphically the results. Keep in mind the following considerations:

- You must include in your report all relevant parameters from the machine where you have executed (processor model, number of cores, main memory size, cache hierarchy, ...) and system software (operating system version, compiler version, ...).
  - You shall use a computer with at least one processor with a minimum of 4 physical cores (excluding *hyperthreading*).
  - Operating system must be Linux.
  - Note that the evaluation for the sequential part and the parallel part must be performed in the same computer and with the same environment conditions.
- Perform every experiment a number of times and take averages. The minimum number of executions per experiment is 10.

Represent in a graph all total execution times obtained from images with different sizes.

Include in your report the conclusions you may draw from the results. Do not limit yourself to simply describing data. You must also look for convincing explanations for the results.

3.3 Parallel version

In this task you will develop the corresponding parallel version using *OpenMP*. In this version both the Gaussian blur and the Sobel operator stages must be parallelized. You must explain in your project report the modifications that you have performed to the sequential code.

Keep in mind the following:

- Any project with a compiler warning will be considered failed.
- Your source code must include among the parameters passed to the compiler, as a minimum, the following: `-std=c++17 -Wall -Wextra -Wno-deprecated -Werror -pedantic -pedantic-errors`, or equivalent parameters.

The project report must include the design decisions that have been taken in order to achieve parallelization. For each decision you must include other considered alternatives and justify the motivation for your choice.

3.4 Evaluation of parallel version

You must repeat evaluations from the first part. Consider varying number of threads from 1 to 16 threads (1, 2, 4, 8 and 16).

Important note: Make sure to compile all the examples with compiler optimizations enabled (compiler options `-O3` and `-DNDEBUG`).

Besides graphs from first section, also represent graphically the obtained speedup.

Include in your report the conclusions you may draw from the results. Do not limit yourself to simply describing data. You must also look for convincing explanations for the results including the impact of the computer architecture.

3.5 Scheduling impact

Perform a study for the cases of 4 threads and 8 threads about the impact from the different OpenMP scheduling models (*static*, *dynamic* y *guided*). Include in your project report the conclusions of the obtained results.

4 Grading

Final grades for this project are obtained considering the following share:

- Total implementation grade: 100%
  - Sequential implementation: 40%
    - Functional tests: 30%
    - Achieved performance: 30%
    - Project report: 40%
  - Parallel implementation: 60%
    - Functional tests: 30%
    - Achieved performance: 30%
    - Project report: 40%

Warnings:

- If the code does not compile, final grade will be 0.
- In case of copies, all implied groups will get a grade of 0.

5 Delivery procedure

All the delivery procedure will be carried out through Aula Global.

Three separate delivery links will be available:

- **Project report delivery**: It contains the project report, that shall be one file in pdf format named **memoria.pdf**.
- **Sequential code delivery**: It contains all the needed source code to compile the sequential version.

  - It must be a compressed file (zip format) named **img-seq.zip**.

- **Parallel code delivery**: It contains all the needed source code to compile the parallel version.
  - It must be a compressed file (zip format) named **img-par.zip**.

The project report shall not exceed 15 pages and it should contain, at least, the following sections:

- **Title page**: it contains the project name, name and NIA of all authors and reduced group number.
- **Table of contents**.
- **Introduction**.
- **Sequential version**: Section explaining implementation and optimization of sequential code. No source code should be included.
- **Parallel version**: Section explaining implementation and optimization of parallel code. No source code should be included.
- **Performance evaluation**: Explanation of different performed evaluations comparing sequential version without optimizations, sequential version with optimizations, and optimized parallel version.
- **Performed tests**: Description of the performed testing plan to ensure the correct execution of both versions.
- **Conclusions**.