

Speed Up of Image Filtering using Parallel Programming Models

José Agustín Barrachina

IEEE Student Member

Instituto Tecnológico de Buenos Aires (ITBA)

Av Eduardo Madero 399, Buenos Aires, Argentina

Email: joseagustin.barra@gmail.com

François Bidet

École Polytechnique

Université Paris-Saclay

Route de Saclay, 91128 Palaiseau, France

Email: francois.bidet@polytechnique.edu

Abstract—The aim of this project was to speed-up a target application through multiple parallelism models (MPI, OpenMP and CUDA) for various architectures (CPU and GPU) and conclude which approach is the most suitable for the specific application.

1. Introduction

A base C/C++ code that implements an image filtering was given. The aim of this work is to optimize the operation time of the code by using parallel programming methods such as MPI, OpenMP and CUDA. The base C/C++ code covers a large spectrum of algorithms in HPC and Big Data. The Image Filtering can be seen as a direct application of Big Data programs. This application uses a stencil-based scheme to apply a filter to an existing image or set of images. It means the main process of this code is a traversal of each pixel of the image and the application of a 2D stencil [1].

2. Image Filtering Program

In this section the general C/C++ algorithm will be explained. A shell script was created in order to compare the program efficiency against numerous gif files with different size and properties like number of images or colors.

2.1. Description

This topic focuses on a specific image filter that is useful to detect objects and edges inside various images. This filter is called Sobel and it applies on a greyscale image. Therefore, the first step is to transform the image from a color one to a set of gray pixels. For this purpose, we will work with GIF image because this format has the following advantages:

- Easy to manipulate
- Widespread format (social networks)
- Allow animation (multiple images)

Then, a blur filter is applied to a small part of the images to exclude the borders. Thus, the main goal is to parallelize an application that apply multiple filters (grayscale, blur and Sobel) to a GIF Image (either a single image or an animated gif). [1]

2.2. Structure

The algorithm can be divided into three main parts:

1. GIF import (section 2.2.1)
2. SOBEL filter (section 2.2.2)
3. Export image (section 2.2.3)

A main function already measures and output the time taken by each of the three listed parts of the project.

2.2.1. GIF import. This part just load the gif file into a pixel array.

2.2.2. SOBEL filter. This part of the code applies actually three different filters:

1. **Gray Filter** Convert the pixels into a grayscale
2. **Blur Filter** Apply blur filter with convergence value
3. **Sobel Filter** Apply Sobel filter on pixels

2.2.3. Export image. This is just the inverse of the first part (section 2.2.1). Converts the array of pixels and creates a gif file which is then stored into a folder.

3. Parallel Optimization Strategies

4. Conclusion

Acknowledgments

References

- [1] P. Carribault and F. Trahay, *INF560 Evaluation*, 3rd ed. <https://www.enseignement.polytechnique.fr/profs/informatique/Francois.Trahay/www/Tprojects-0.html>