Universidade de Aveiro

Mestrado Integrado em Engenharia Computacional

Computação Paralela

Lesson 1: Parallelization by Multithreading

Academic year 2019/2020 Nuno Lau/Manuel Barroso

Parallelization of a given program may be achieved by partitioning the problem in different sections that may be executed in parallel by using threads. In this lesson, we will use the pthreads library in order to speedup the process of linear filtering of images.

Images considered in this lesson will be monochromatic (gray scale), where each pixel's luminance is represented by an integer value (0..255). The value of 0 represents a black pixel, while the value of 255 represents a white pixel.

The use of filters to process images is very common and can be used to obtain very different effects. In this lesson, only linear filters will be used, in which the value obtained in the filtered image is a linear combination of the values of the nearby pixels in the original image. In the general case of a 2D image and filter, the linear filtering operation is similar to a convolution between the image matrix and the filter matrix. Considering an image I and a filter f where the filter has width fw and height fh, the value of each pixel of the filtered image I' can be obtained through:

$$I'(x,y)=\sum_{u=0}^{fw}\sum_{v=0}^{fh} I(x-fw/2+u,y-fh/2+v)\times f(u,v)$$

The objective of this lesson is to, starting from a sequential implementation of the image linear filtering (provided by the `cp_imageFilter.tgz`), develop image filters implementations that are parallelized using the pthreads library. To accomplish this work, you may consider the following phases.

1. Develop a parallelized versions of the linear filtering by:
   1.1. cutting the image in horizontal strips and assigning the filtering of each strip to a different thread. The number of threads should be controlled by the new `-t nthreads` option.
   1.2. cutting the image in vertical strips and assigning the filtering of each strip to a different thread
   1.3. by considering that successive pixels in each line will be processed by different successive threads.
2. Determine the speedups that may be achieved for each of the previous implementations for several different conditions. In order to get more meaningful and stable speedups it may be worth to repeat each filtering operation several times (ex: 100 times), so that the measures of elapsed time get more precise. The speedups should be checked when:
   2.1. The number of threads changes
   2.2. The size of the image changes
   2.3. The size of the filter changes

2.4. The different compiler optimization levels are activated

Bibliography

[1] POSIX Thread Programming, Blaise Barney, Lawrence Livermore National Laboratory, https://computing.llnl.gov/tutorials/pthreads/