Glenn Contreras

Dr. Pallipuram

High Performance Computing

23 February 2017

Lab Report

Abstract

This report discusses the design and implementation of program 2 and the results of the speedup and parallel efficiency. The goal of this program was to convolve an image of various sizes into horizontal and vertical gradients using the Gaussian distribution function and its derivative.

Introduction

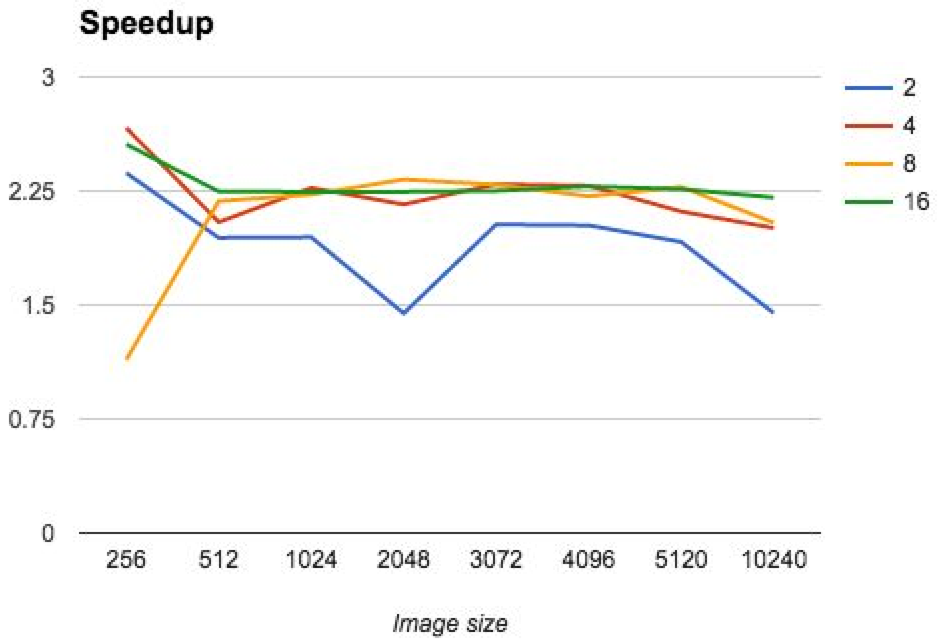
The goal of this lab is to write a serial and parallel program that takes in a single image and outputs its horizontal and vertical gradients. Then analyze the speedup and parallel efficiency of both programs to see how parallel computing effects computationally intensive algorithms. The horizontal and vertical gradients were calculated using the Gaussian distribution formula and its derivative. The parallel and serial time were then recorded to calculate the speedup and efficiency of the program.

Parallelization Methodology

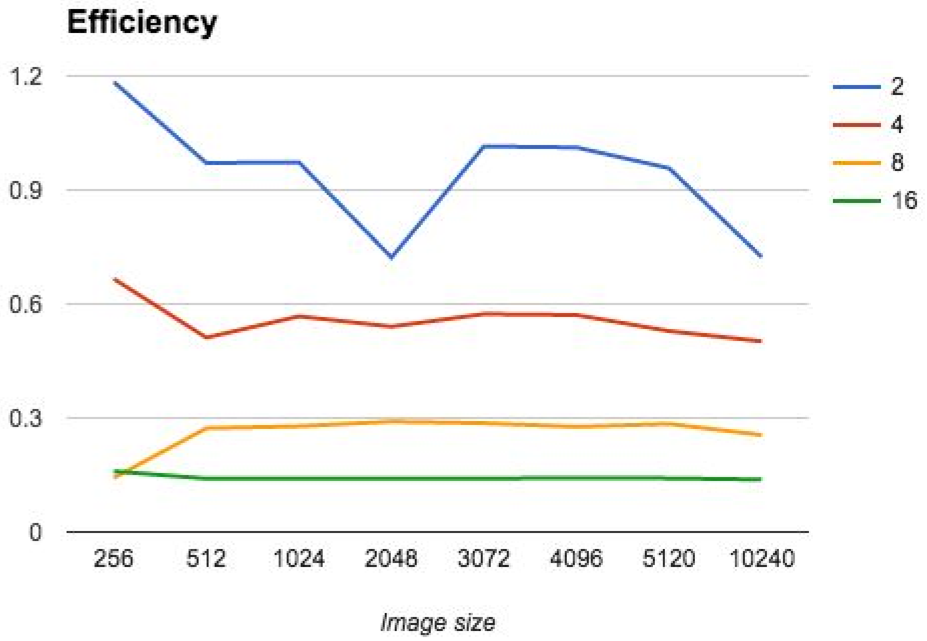
The image was partitioned by dividing the number of processor by the image size. This also determined the offset used by the convolution function. Each thread calculated a unique offset from its id. It then convolves the image with the gaussian distribution function and saves it in a temporary array which it then uses to calculate the horizontal gradient with the Gaussian derivative. This process is then repeated in a similar fashion to obtain the vertical gradient. After the gradients have been calculated, the threads terminate and the main function writes the image to a file.

Results

The average speedup time of the program is 2.123. As you can see from the chart below. The program has weak scalability since the speedup does not increase with respect to processor count. Although there is noticeable difference in from one thread to two threads for when 8 processors were used denoted by the yellow line. This could be due to my computer freeing up processing time.



The average parallel efficiency came out to be 0.4766 with the efficiency greatly suffering when using more threads as shown in the graph below.


Efficiency
1.2
0.9
0.6
0.3
0
256
512
1024
2048
3072
4096
5120
10240
Image size
2
4
8
16