CS-206 HW2

Performance and efficiency

1 Amdahl's Law

1. Tianhe-2 is currently the most powerful supercomputer with 3'120'000 cores. Consider a program where 0.01% of the runtime is not parallelizable. Assuming the program performance is the same on all of those cores and there are no additional overheads,

   (a) What is the parallel speedup on 30, 300, 3'000, 30'000, 300'000 and 3'000'000 cores?

   (b) If we provision the application with an infinite amount of cores, what would be the speedup?

   (c) What is the number of cores that provides 95% of the speedup calculated in (b)?

2. Amdahl's law indicates there is little point in using large-scale systems with millions of cores to accelerate programs that have even a small fraction of non-parallelizable work (which is often inevitable, e.g., reading input data). Why do people build such systems?

2 Parallel Image Processing

The sobel algorithm is widely used in image processing applications to detect edges in images. It calculates the derivatives of an image in the vertical and horizontal directions. Each derivative is calculated by convolution its respective kernel with the image. The kernels for the vertical (K_y), and horizontal (K_x) directions are presented below.

$$K_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \quad K_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

The convolution G of a kernel with an image is computed by a loop that calculates the value of G(x,y) for each pixel in the input image, as shown below.

$$G(x,y) = \sum_{i=0}^{2} \sum_{j=0}^{2} P(x+i-1, y+j-1) * K(i,j)$$

In the equation, P(x,y) is the pixel value with coordinates (x,y) and K(i,j) is the kernel value with coordinates (i,j).

The result of the convolutions are two matrices of derivatives with the same size as the image. The derivatives can be used to plot the edges of the input image. As an approximation, we can say that a pixel P(x,y) belongs to an edge if $|G_x(x,y)| + |G_y(x,y)| > t$ where G_x and G_y are the horizontal and vertical derivatives and t is a given threshold.

To plot the edge image, we set all edge pixels to 255 (white) and all other pixels to 0(black).

The algorithm pseudocode is presented below:

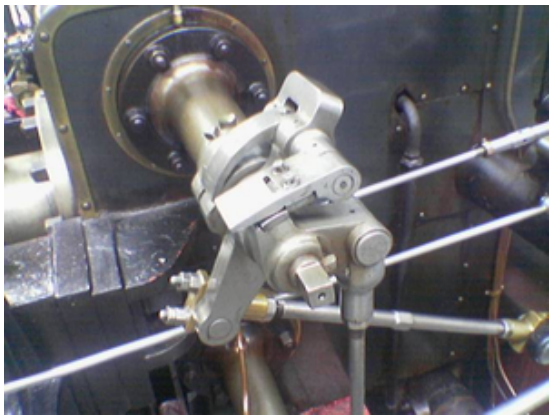
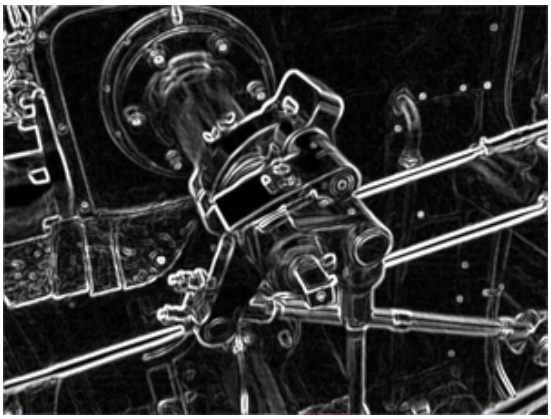

Figure 1: Example of sobel application: on the left we have the original image and the edge image is presented on the right

```
for y:=0 to imageHeight-1 begin
  for x:=0 to imageWidth-1  begin
    convResultX :=0
    convResultY :=0
    for i:=0 to 2 begin
      for j:=0 to 2 begin
        // Clamp the image coordinates
        clampedX = (x+i-1 < 0)? 0 :
                     ( (x+i-1 > imageWidth-1)? imageWidth-1 : x+i-1)
        clampedY = (y+j-1 < 0)? 0 :
                     ( (y+j-1 > imageHeight-1)? imageHeight-1 : y+j-1)

        // Do the actual calculation
        convResultX += inputImage(clampedX,clampedY)*kernelX(i,j)
        convResultY += inputImage(clampedX,clampedY)*kernelY(i,j)
      end for
    end for

    if( abs(convResultX) + abs(convResultY) > threshold ) then
      outputImage(x,y) = 255
    else
      outputImage(x,y) = 0
    end if
  end for
end for
```

Figure 1 presents an example of the application of the described algorithm.

Given a processor with N cores, how would you parallelize this algorithm in order to achieve best performance. Ignore any cache effects.

Write the pseudo code for one thread, explicitly indicating its inputs and outputs. Briefly describe how your parallelized version of the algorithm is supposed to run (i.e., when and how threads are going to be started, if there is any additional work to be done to set up parallelization).

3 Submission

Deadline: 10.03.2015

Please make a PDF file of your answer and upload it in the moodle using related box. Do not forget to write your Name and Sciper number.

Please write clearly and concisely.