Laboration 1 - TDDC78

Jonathan Karlsson - jonka293 - 890201-1991
Niclas Olofsson - nicol271 - 900904-5338

17 maj 2013

1 Program description

1.1 Threshold filter

This program calculates the average intensity of the image and makes all pixels with a higher-than-average value white, and all other pixels black.

The whole image is read on the root node. The root node also calculates the average intensity of the image. The calculated intensity is sent to all other nodes via MPI broadcast.

The image data array is split in as many parts as there are nodes, and each node gets its own part via MPI scatter. Since each node needs to know the data length to receive, this is done in two steps where the data length is sent via MPI broadcast, and then sent with MPI scatter. Each node then runs the threshold filter on its own part of the image. MPI gather then reassembles the resulting image, which is written to disk by the root node.

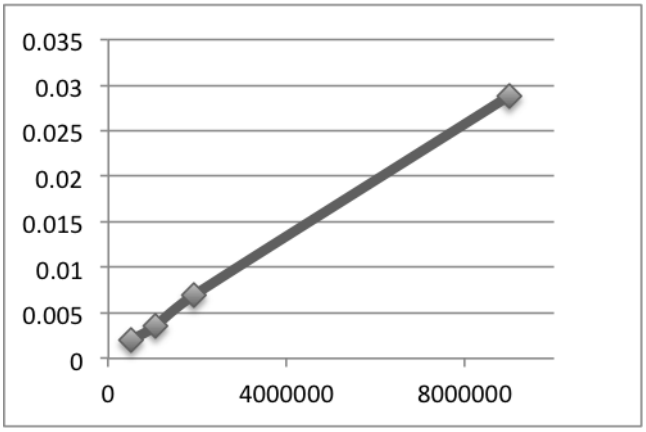
1.2 Blur filter

For each pixel this program calculates the average color of the neighbors with a given radius, of the pixel and sets the pixels color to the average. This gives a blurred effect over the whole image with a small radius giving almost the same image back and a big radius gives a very blurry image.

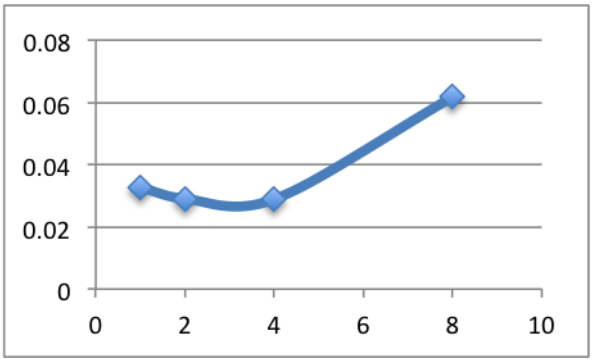
The whole image is read on the root node and then the intervals that each processor is going to work in is calculated and transmitted with scatterv. The reason that scatterv is necessary is because there's a lot of dependencies between each chunk of the image so each processor need a bit more data so it can read all neighbors, otherwise there will be some broken lines in the image. Now each chunk is iterated by the blur filter and then transmitted to the root node with gatherv, the root node must take special care to get rid of some overhead created by the sending of extra data. This is done by pointing a bit forward in the array storing the image.

2 Execution times

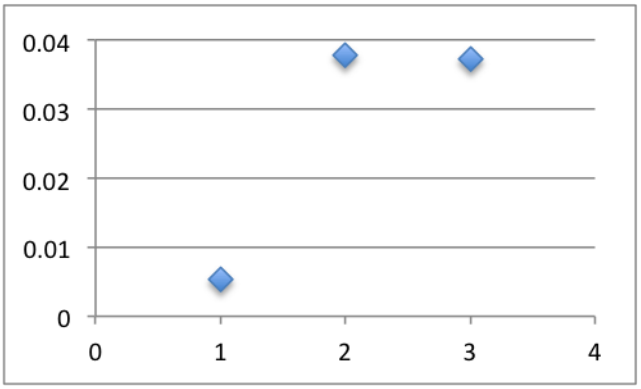
Below are figures showing the execution times, some comments is that the thresh filter scales poorly since it's such a small problem and the overhead of communication is too big. The blur filter scales better but we notice some overhead when we need to send to different processors to access our nodes.



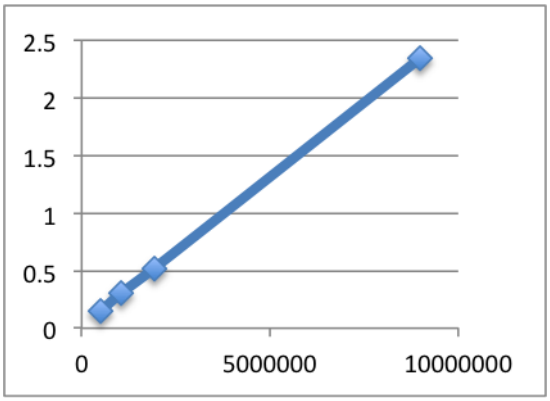
Figur 1: Using the same amount of nodes on differente images. (Thresh)



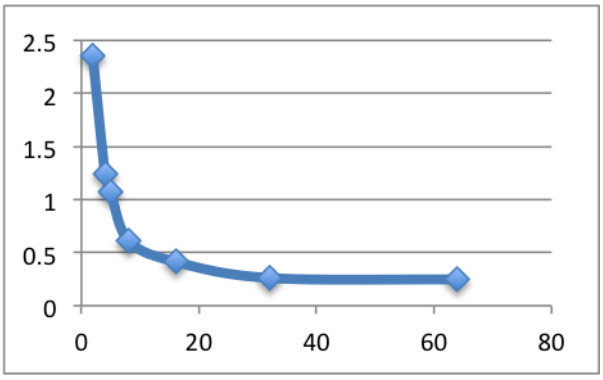
Figur 2: Using the same image with different amount of modes. (Thresh)



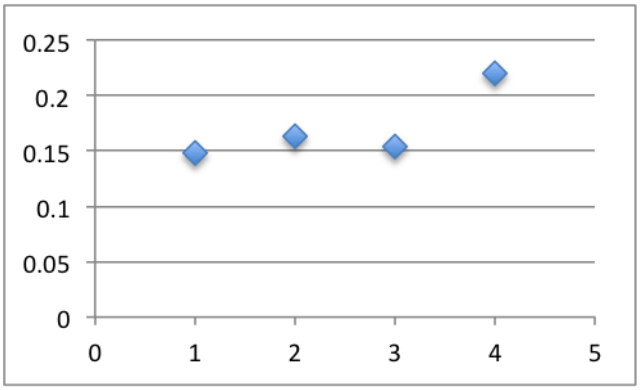
Figur 3: Scaling the image at the same rate as the number of nodes. (Thresh)



Figur 4: Using the same amount of nodes on differente images. (Blur)



Figur 5: Using the same image with different amount of modes. (Blur)



Figur 6: Scaling the image at the same rate as the number of nodes. (Blur)