Authors: Filip Källström filka786; Emil Lindberg emili903 Date: 2013-05-20

Report lab 1 – Image Filter (MPI)

Description

Blur

Each process reads the image and is assigned a range of rows that it is supposed to blur. Each process then just applies the given algortihm to blur their assigned rows with the given radius. We don't need any communication at this stage since all processes have access to the full image and can therefore get all of the information it needs from the surrounding pixels. After the blur we call "MPI_Gatherv" to collect each process's range of rows together to a fully blurred image.

The only communication needed with this approach is at the end when each process have to send their blurred part of the image to process 0 so it can write the fully blurred image.

Threshold

We start by letting the process with ID 0 read the image file and then broadcast the image size to the rest of the processes.

After this we calculate displacements for each process and how much data each process will work with to prepare for a scatter of the image file using "MPI_Scatterv". We let each process get the same amount of pixels to work with. In the cases where this is not possible we distribute the rest of the pixels to as many processes as possible.

With the image scattered each process can now calculate the sum of the pixel values for their part of the image and then we do a "MPI_Allreduce" call with the sum operation to get the sum of the whole image.

Since we sent the size of the image earlier to all processes each process can now calculate the average intensity of the whole image and threshold their part of the image.

At the end we again use the earlier calculated displacements and work data amount to do a "MPI_Gatherv" call to get the final image.

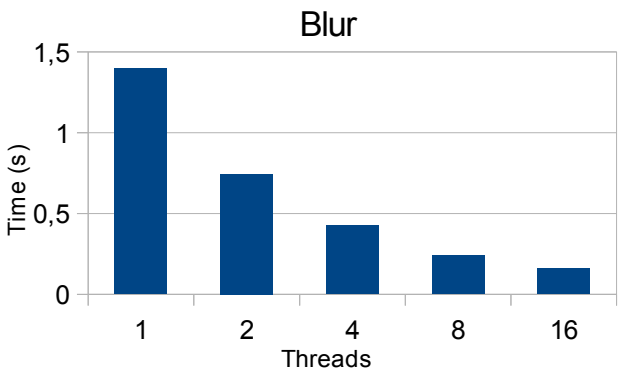
Execution times

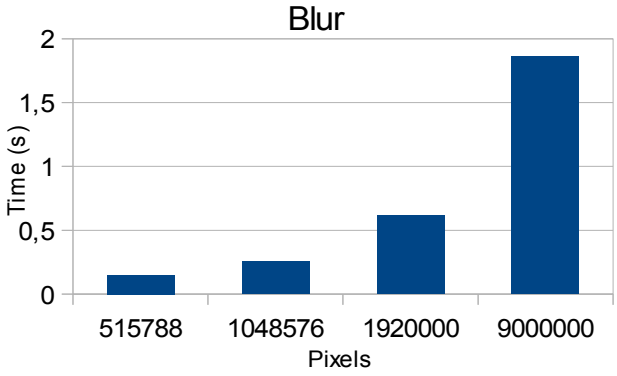
The data for the diagrams below is averages of 5 runs and can be seen in the appendix at the end of this report.

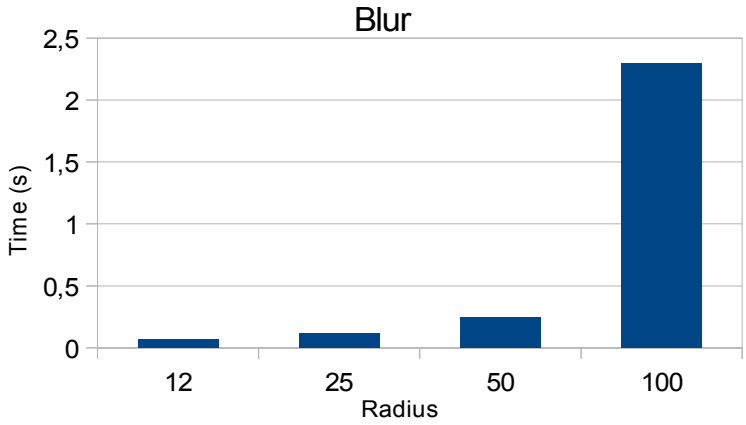
Default values for the variables if they are not changed in the diagrams are:

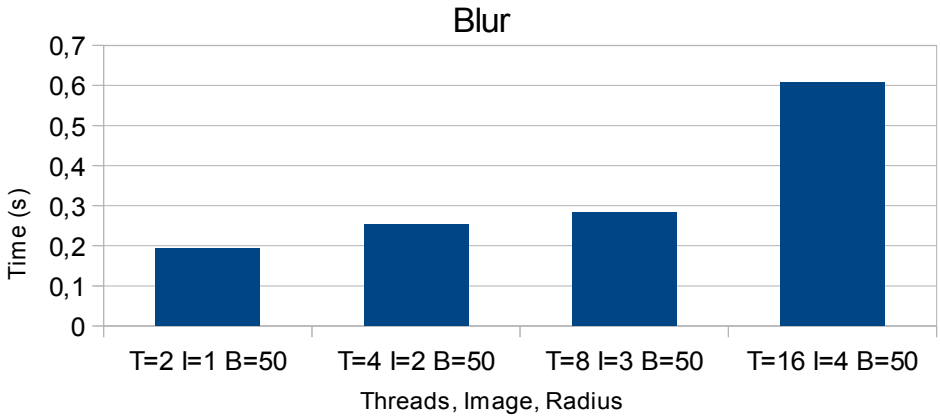
- Threads: 8 (except for the first diagram below, which is run with 4 threads)
- Image 3: 1920000 pixels
- Radius: 50

Blur





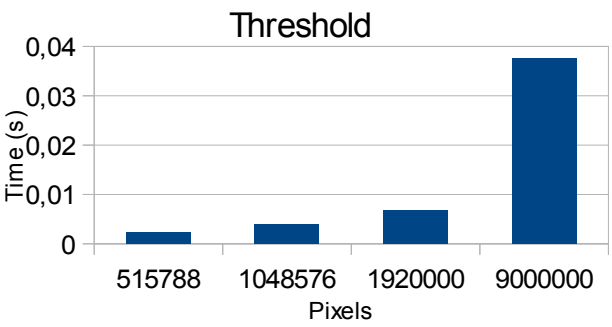


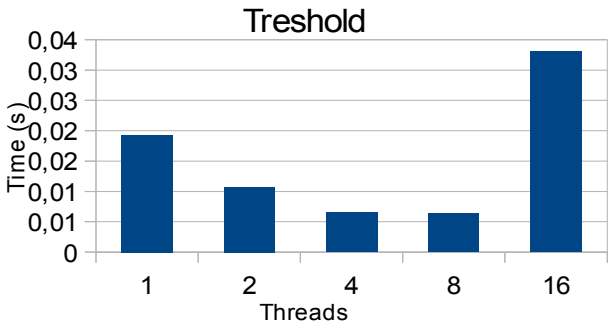


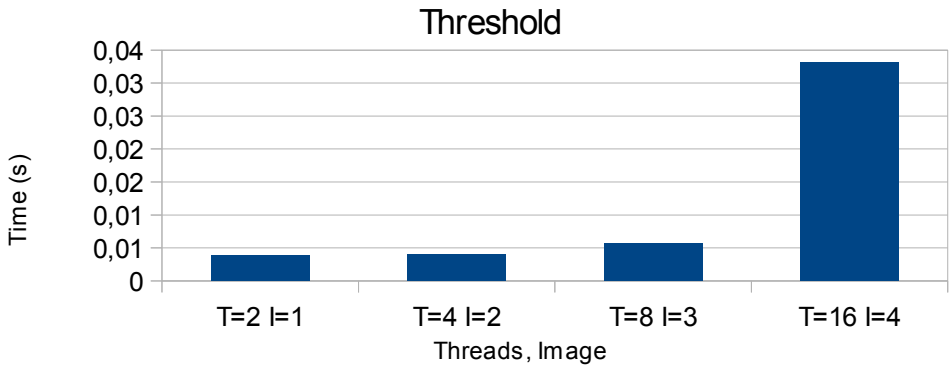
What's worth noting is that the last measure in the radius diagram had a very large impact on the execution time. The reason that the execution time increases by such a large amount by doubling the radius from 50 to 100 could probably be caused by an increased amount of cache misses in every calculation of the blur filter since we basically have to include 50 more rows above and below each pixel in the calculations of the blur filter.

The other diagrams seem to look as expected. The time is decreased to half when doubling the number of threads. Time taken when increasing the picture size is also increased as expected.

Threshold







Here it's worth to note the diagram where we increase the number of threads. We can see a normal behavior of time decreasing to half for every doubling of number of threads. But when we reach 16 threads the time suddenly takes much longer time. Our thoughts on this is that the communication needed in the threshold filter probably gets a little bit to complex with that amount of threads and therefore makes a big impact on the execution time.

The last diagram shows expected behavior the first 3 runs have close to the same run time since the number of pixels is about doubled and the number of threads is also doubled. In the last run the image is almost 5 times as big so that have a big impact on the run time but more importantly it's run on 16 threads which can be seen in the second diagram doesn't seem to have good speed for our program.

Authors: Filip Källström filka786; Emil Lindberg emili903 Date: 2013-05-20

Flops

Blur

The blur algorithm have 2 loops where both loop of all pixels and have an inner loop that loops over the radius. There is 18 flops that is performed for all pixels then there is 28 flops in the inner most loop which also loops over the radius. This leads to the following equation.

$$2*(numPixels*18+numPixels*radius*28)$$

Threshold

The threshold filter algorithm has 2 parts. First it sums up all red green and blue values for each pixel to a total sum of the pixel. This is 3 times the number of pixels flops. Then we have to check for each pixel if it's supposed to be black or white which is again a sum of the red green and blue value of the pixel, that is 2 times number of pixels flops. All in all we get the following number of flops.

$$3*numPixels+2*numPixels=5\text{numPixels}$$

Calculation of MFLOPS

Calculation of MFLOPS with an image size of 1920000 pixels and execution time of 0.0068395 seconds.

$$totalFLOPS=5*1920000=9600000$$

$$\frac{9600000\,FLOPS}{0.0068395\,sec}=1403.6\,MFLOP/s$$

Authors: Filip Källström filka786; Emil Lindberg emili903 Date: 2013-05-20

Appendix:

Blur

Threads	Time
1	1,398
2	0,743
4	0,425
8	0,241
16	0,159

Image	Time
515788	0,151
1048576	0,252
1920000	0,614
9000000	1,86

Radius	Time
12	0,066
25	0,115
50	0,248
100	2,298

All	Time
T=2 I=1 B=50	0,193
T=4 I=2 B=50	0,254
T=8 I=3 B=50	0,282
T=16 I=4 B=50	0,607

Threshold

Threads	Time
1	0,0193206
2	0,0107289
4	0,006587
8	0,006325
16	0,033007

Image	Time
515788	0,0023445
1048576	0,00389667
1920000	0,0068395
9000000	0,0375821

All	Time
T=2 I=1	0,00385612
T=4 I=2	0,0040474
T=8 I=3	0,005692
T=16 I=4	0,0331243