

GPU and GPGPU Programming

Shuai Lu 170742

Assignment 5

Task

1. Matrix, Vector Operations and Reduction

You have to program the matrix and vector operations (matrix-vector multiplication, vector-vector operations, and reduction) on the CPU. 1.b) Implement the same operations on the GPU.

Find the comments "TASK" in the code and implement them (one CPU version and one GPU version). For testing a small hard-coded 4x4 matrix, and .txt files containing sparse matrices are provided.

2. Improve the performance of the GPU implementation You must use shared memory for the vector in the matrix-vector-multiplication! For the reduction you must use shared memory as well (as discussed in the lecture). Measure the performance of CPU, unoptimized (without shared memory) and optimized (with shared memory) GPU versions.

3. Test using Image Deblurring Method Once the Conjugate Gradient method is working for the matrices it can be used to solve practical applications. A simple image de-blurring application is implemented that works for a known blurring filter as described in the following: An image is loaded and blurred with a filter kernel. The image can be de-blurred again by solving the equation $Ax = b$ where x is the unknown input image, that was filtered with filter operation A such that the result is the known blurred image b . In order to formulate the blurring operation (convolution) as a matrix multiplication, b and x are represented as vectors and each row of matrix A is one filter kernel for the whole image x . This naive method is very memory inefficient ($O(N^2)$ where N is the number of pixels). Your task is to test if your GPU implementations (of tasks 1. and 2.) work and to measure the performance. Hint: Because of numerical inaccuracies for the very large matrices that occur you will have to increase the error tolerance for the CG-method to terminate.

BONUS:

1. Implement a sparse matrix format. 2. Modify the code to divide the image into small patches and solve for them separately. This will allow to run the de-blurring on larger input images.

Outcome:

The matrix-vector multiplication, vector-vector operations, and vector reduction are implemented both in CPU and GPU. The shared memory is used in the GPU implementation (the matrix-vector-multiplication and vector reduction) to improve the performance. The result of CG method when matrix size equal to 16 is shown in Fig.1.

The measurement of the run time of CPU, GPU and optimized GPU implementations are shown in Fig.2.

The GPU implementations also work for image deblurring. The outcome is shown in Fig.3. And the performance is shown in Fig.4.

```

Microsoft Visual Studio Debug Console
Device(s): 1
16 16 256
File read successfully
1 16 16
File read successfully
iteration #0, with rho_cpu = 16397.6
iteration #1, with rho_cpu = 4.14948
iteration #2, with rho_cpu = 0.654237
iteration #3, with rho_cpu = 0.0411956
iteration #4, with rho_cpu = 0.0523924
iteration #5, with rho_cpu = 0.554485
iteration #6, with rho_cpu = 0.0119661
iteration #7, with rho_cpu = 0.00540168
iteration #8, with rho_cpu = 0.00242768
iteration #9, with rho_cpu = 0.00185741
iteration #10, with rho_cpu = 0.00374835
iteration #11, with rho_cpu = 0.000319326
iteration #12, with rho_cpu = 6.10132e-05
iteration #13, with rho_cpu = 0.000122965
iteration #14, with rho_cpu = 0.00132788
iteration #15, with rho_cpu = 0.000428977
Solution found at iteration #16, with rho = 0.000123

minrho was 0.000061
CPU elapsed time: 33.1028ms
iteration #0, with rho_gpu = 16397.6
iteration #1, with rho_gpu = 4.14949
iteration #2, with rho_gpu = 0.65424
iteration #3, with rho_gpu = 0.0411981
iteration #4, with rho_gpu = 0.0576251
iteration #5, with rho_gpu = 0.313966
iteration #6, with rho_gpu = 0.0119163
iteration #7, with rho_gpu = 0.00640181
iteration #8, with rho_gpu = 0.00242819
iteration #9, with rho_gpu = 0.00877144
iteration #10, with rho_gpu = 0.00226459
iteration #11, with rho_gpu = 0.000295746
iteration #12, with rho_gpu = 6.10026e-05
iteration #13, with rho_gpu = 0.00021349
iteration #14, with rho_gpu = 0.000287313
iteration #15, with rho_gpu = 0.000414402
Solution found at iteration #16, with rho = 0.000867

minrho was 0.000061
GPU elapsed time: 296.364ms
CPU
errors:
sum | 0.0367317
avgs | 0.00229573
max | 0.00563431

GPU
errors:
sum | 0.0985584
avgs | 0.0061599
max | 0.015625

```

Figure 1: MatrixSet=16

MatrixSet=16	Time	Error_sum	Error_avgs	Error_max
CPU	34	0.0367317	0.0022957	0.00563431
GPU	309	0.0985584	0.0061599	0.015625
OptimizedGPU	258	0.103142	0.0064464	0.0177574
MatrixSet=64	Time	Error_sum	Error_avgs	Error_max
CPU	189	0.021759	0.00034	0.00128174
GPU	497	0.0204773	0.00032	0.00115967
OptimizedGPU	401	0.0295715	0.0004621	0.00195312
MatrixSet=200	Time	Error_sum	Error_avgs	Error_max
CPU	647	0.244141	0.0012207	0.00537109
GPU	1074	0.241699	0.0012085	0.00390625
OptimizedGPU	964	0.241699	0.0012085	0.00390625

Figure 2: Measurement of different implementations on different matrix sizes

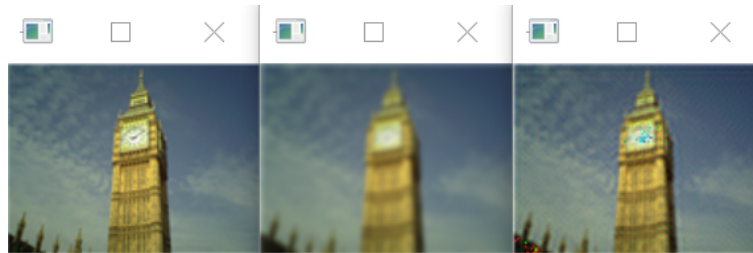


Figure 3: DeBlurring

```
C:\CS380\cs380-2021\5_assignment\Debug\assignment.exe
Device(s): 1
computing red channel
iteration #0, with rho_gpu = 1.3567e+08
iteration #1, with rho_gpu = 245085
iteration #2, with rho_gpu = 48818.8
iteration #3, with rho_gpu = 12950.5
iteration #4, with rho_gpu = 13741.1
iteration #5, with rho_gpu = 75720.2
iteration #6, with rho_gpu = 284494
iteration #7, with rho_gpu = 15499.6
iteration #8, with rho_gpu = 10461.6
Solution found at iteration #8, with rho = 11791.244141
minrho was 10461.625977
errors:
sum | 9095.42
avgs | 0.740188
max | 6.13171
computing green channel
iteration #0, with rho_gpu = 1.61617e+08
iteration #1, with rho_gpu = 1.76167e+06
iteration #2, with rho_gpu = 47150
iteration #3, with rho_gpu = 12984.7
iteration #4, with rho_gpu = 12258.1
Solution found at iteration #4, with rho = 12258.071289
minrho was 12258.071289
errors:
sum | 10832.8
avgs | 0.881575
max | 5.60288
computing blue channel
iteration #0, with rho_gpu = 1.34198e+08
iteration #1, with rho_gpu = 706081
iteration #2, with rho_gpu = 20059.1
iteration #3, with rho_gpu = 6722.78
Solution found at iteration #3, with rho = 7883.365234
minrho was 6722.778809
errors:
sum | 7539.15
avgs | 0.613537
max | 7.55138
```

Figure 4: Performance