CSE 5441: Program 4 Report

Harsh Gupta

SECTION : 2:20 pm – 3:55pm Tu Thu

Contents

Objective

The objective of this lab is to use CUDA and analyse how serial execution of the two different programs differs from the parallel version of it. Two different versions of programs were implemented and analysed. They are as follows

a. Matrix multiplication with its transpose
b. Sobel transformation

Both the programs executed the same operations first serially then parallelly. The goal was to contrast the performance of serial and parallel program, which in this case means CPU vs GPU. CUDA api's provided by NVidia are straight forward and easy to understand. Selecting dimensions of Grid and Blocks as well as the work load distribution among the threads was a bit complicated task. To imagine a 2D array as a linear array and then assigning work load distribution over it was a bit tricky.

Parallelizing the code - CUDA organization (grid, block, thread)

Matrix Multiplication

Assumption:
Size of the matrix = 1024 * 1024

In the serial version, we have three nested loops to calculate the output matrix from the input matrices. To do away with two of those nested matrices I decided to have 1024 threads which will help to parallelize the code. But two nested matrices accounts for 1024 * 1024 operations, but we only have 1024 threads as it is a limitation on the side of OSC. So, each thread will perform 1024 computations, to compensate. Therefore, each thread will end up calculating a cell of the resultant matrix. Dimensions used are the once that provided maximum performance by trying different values. 2D representation is used for the program.

Number of grids = 1
Number of blocks = (32, 32) = 1024 blocks per grid
Number of thread per block = (32, 32) = 1024
Total number of threads = 1024

Sobel transform

This program also distributes work load like the part1 program. The only difference here is that the matrix of part1 is replaced by a 2D bmp image and we perform Sobel operation as per the provided code. Here again I used a 2D blocks and threads in order to have thebest performance. Dimensions used are the once that provided maximum performance by trying different values.

Number of grids = 1
Number of thread per block = (32, 32)
Number of blocks = (Height of Image/32 + 1, Width of Image/32 + 1)
Total number of threads = Depends on the dimension of the image

There is one added to the x and y value of blocks to take care of images which are not multiples of 32. This can be problematic for cases where the thread coordinates are calculated for values beyond the size of the image but that case is taken care of in the device.

Summary of execution time

Matrix Multiplication

Program	Matrix Dimensions	Time (seconds)	Gflops
Serial	**1024 * 1024**	**6.09673**	**0.352236**
Cuda	**1024 * 1024**	**0.28795**	**7.457908**

Gflops = (Total number of floating point operations/second)/10^9
Serial GFLOPS = (2*(1024^3)/3.09671)/10^9 = **0.352236** GFLOPS
Cuda GFLOPS = (2*(1024^3)/0.27283)/10^9 = **7.457908** GFLOPS

The CUDA program is almost 21 times faster.

To compute time for CUDA operations, I included the time to allocate memory as well. This consumed considerable amount of time. The time taken only at kernel was very less and would have given 100 times faster result than serial result.

Sobel transform

Threshold remained same for serial and CUDA versions. Therefore, making a single column for threshold.

Time is given in seconds without considering malloc and memcpy for CUDA

Image	Serial Time	Parallel Time	Threshold	Speed Up
Coins.bmp	**0.153040**	**0.001452**	**49**	**105x**
Test_sample_1.bmp	**2.258575**	**0.010831**	**36**	**208x**
Test_sample_2.bmp	**3.551521**	**0.016444**	**54**	**216x**
Test_sample_3.bmp	**10.005480**	**0.044006**	**39**	**227x**
Test_sample_4.bmp	**2.710359**	**0.012838**	**42**	**212x**

Time for considering cuda malloc and memcpy as well

Image	Serial Time	Parallel Time	Threshold	Speed Up
Coins.bmp	**0.153795**	**0.276493**	**49**	**0.55x**
Test_sample_1.bmp	**2.208026**	**0.287067**	**36**	**7x**
Test_sample_2.bmp	**3.537736**	**0.294186**	**54**	**12x**
Test_sample_3.bmp	**10.050755**	**0.348766**	**39**	**29x**
Test_sample_4.bmp	**2.712026**	**0.285489**	**42**	**9x**

Analysis and Observations

Did this program perform better sequentially or in parallel?

- **Matrix Multiplication**

There is a 21x improvement when using GPU. This is an amazing performance improvement, which was surprising.

- **Sobel Transformation**

Without considering memcpy and cuda malloc
There is almost 100x improvement when using GPU. This is even more surprising than Matrix multiplication. Here for time computation I have only considered the time to compute the edge detection and not the time for memory allocation. Another reason for such an improvement is doing atomic addition on device and not doing any computations on the main to update percent_black_cells.

This increase in sobel transformation is even higher (more than 200x) for images of larger number of pixels, this is essentially because higher the number of pixel more is the parallelization in a 2D format I have chosen.

Considering memcpy and cuda malloc
When we are considering both memcpy and cuda malloc then speed up compare to it's serial version is not that significant as expected. This is essentially because of the overhead of data transfer for images. More surprisingly for a smaller image like coins.bmp there is no speed up only rather it is slower than the serial version. This was not expected but the possible reason is that the overhead for data transfer is more than the amount of work which is done in the kernel.

For matrix multiplication even after considering the malloc and memcpy the code is running 21 times faster and was running way faster if only the kernel execution was concerned. Hence I gave the results only for the former.

CUDA (and GPU programming) are amazing and very powerful tools. It is fast, efficient but may be a bit tricky to code. This execution timings from this lab clearly displays the power of a GPU and is importance in High performance computing. The Sobel image transform on Cuda made me realize the role of GPU in gaming, where 30-60 frames must be rendered per second. Without a GPU's large-scale parallelization, it would be almost impossible to run a high graphic demanding game.

Results

```
-bash-4.2$ ./lab4p1
************************************************
Matrix multiply on Host(CPU)
Time taken for matrix multiplication on CPU (sec) = 6.09673
GFLOPS/sec in Host = 0.352236
************************************************

************************************************
Matrix multiply on Device(GPU)
Time taken for matrix multiplication transpose in GPU with 32 block(s) of 32   threads (sec) = 0.28795
GFLOPS/sec in Device = 7.457908
************************************************



-bash-4.2$ ./lab4p2 coins.bmp 1.bmp 2.bmp
Image Info ::
      Height=246 Width=300
**************************************************************************
SOBEL EDGE DETECTION
**************************************************************************
**************************************************************************
Serial Execution
Elapsed time for Sobel Operation time : 0.153040
Threshold: 49
**************************************************************************
**************************************************************************
CUDA Execution
Elapsed time for Sobel Operation time : 0.001452
Threshold: 49
**************************************************************************
-bash-4.2$ ./lab4p2 test_sample_1.bmp 1.bmp 2.bmp
Image Info ::
      Height=1080 Width=1920
**************************************************************************
SOBEL EDGE DETECTION
**************************************************************************
**************************************************************************
Serial Execution
Elapsed time for Sobel Operation time : 2.258575
Threshold: 36
**************************************************************************
**************************************************************************
CUDA Execution
```

```
Elapsed time for Sobel Operation time : 0.010831
Threshold: 36
********************************************************************************
-bash-4.2$ ./lab4p2 test_sample_2.bmp 1.bmp 2.bmp
Image Info ::
	Height=1080 Width=1920
********************************************************************************
SOBEL EDGE DETECTION
********************************************************************************
********************************************************************************
Serial Execution
Elapsed time for Sobel Operation time : 3.551521
Threshold: 54
********************************************************************************
********************************************************************************
CUDA Execution
Elapsed time for Sobel Operation time : 0.016444
Threshold: 54
********************************************************************************
-bash-4.2$ ./lab4p2 test_sample_3.bmp 1.bmp 2.bmp
Image Info ::
	Height=2160 Width=3840
********************************************************************************
SOBEL EDGE DETECTION
********************************************************************************
********************************************************************************
Serial Execution
Elapsed time for Sobel Operation time : 10.005480
Threshold: 39
********************************************************************************
********************************************************************************
CUDA Execution
Elapsed time for Sobel Operation time : 0.044006
Threshold: 39
********************************************************************************
-bash-4.2$ ./lab4p2 test_sample_4.bmp 1.bmp 2.bmp
Image Info ::
	Height=1080 Width=1920
********************************************************************************
SOBEL EDGE DETECTION
********************************************************************************
********************************************************************************
Serial Execution
Elapsed time for Sobel Operation time : 2.710359
Threshold: 42
********************************************************************************
********************************************************************************
```

CUDA Execution
Elapsed time for Sobel Operation time : 0.012838
Threshold: 42
****

```
Image Info ::
         Height=246 Width=300
**********
SOBEL EDGE DETECTION
**********
**********
Serial Execution
Elapsed time for Sobel Operation time : 0.153795
Threshold: 49
**********
**********
CUDA Execution
Elapsed time for Sobel Operation time : 0.276493
Threshold: 49
**********
-bash-4.2$ ./lab4p2 test_sample_1.bmp 1.bmp 2.bmp
Image Info ::
         Height=1080 Width=1920
**********
SOBEL EDGE DETECTION
**********
**********
Serial Execution
Elapsed time for Sobel Operation time : 2.208026
Threshold: 36
**********
**********
CUDA Execution
Elapsed time for Sobel Operation time : 0.287067
Threshold: 36
**********
-bash-4.2$ ./lab4p2 test_sample_2.bmp 1.bmp 2.bmp
Image Info ::
         Height=1080 Width=1920
**********
SOBEL EDGE DETECTION
**********
**********
Serial Execution
Elapsed time for Sobel Operation time : 3.537736
Threshold: 54
**********
**********
CUDA Execution
Elapsed time for Sobel Operation time : 0.294186
Threshold: 54
**********

-bash-4.2$ ./lab4p2 test_sample_3.bmp 1.bmp 2.bmp
Image Info ::
         Height=2160 Width=3840
**********
SOBEL EDGE DETECTION
```

```
**************************************************************************
**************************************************************************
Serial Execution
Elapsed time for Sobel Operation time : 10.050755
Threshold: 39
**************************************************************************
**************************************************************************
CUDA Execution
Elapsed time for Sobel Operation time : 0.348766
Threshold: 39
**************************************************************************
-bash-4.2$ ./lab4p2 test_sample_4.bmp 1.bmp 2.bmp
Image Info ::
         Height=1080 Width=1920
**************************************************************************
SOBEL EDGE DETECTION
**************************************************************************
**************************************************************************
Serial Execution
Elapsed time for Sobel Operation time : 2.712026
Threshold: 42
**************************************************************************
**************************************************************************
CUDA Execution
Elapsed time for Sobel Operation time : 0.285489
Threshold: 42
**************************************************************************
```