Hassan Ahmad

ID: 59014371

WES-237B Lab 4

Lab 4 Deliverables

Part 1:

`ex1.cu` Description:

This example defines a CUDA function called *myKernel* that takes a pointer as an input. *myKernel* prints various information about the parallel processes: block number, thread number, thread index value, and the corresponding values of an input array. The example's main function defines a vector of 4 integers and allocates GPU memory that is 4 times the size of that vector. Then, the array is copied from the CPU memory into the GPU memory, and *myKernel* is called to print the GPU processing information. The main function waits for all the processes of the GPU to finish with the *cudaDeviceSynchronize()* function, then lastly frees the memory in the GPU.

`ex1.cu` vs `ex2.cu` :

The key difference between ex1.cu and ex2.cu is how we have selected the block and thread scheme for the GPU to receive the input data. Ex1.cu uses two blocks with two threads per block, and ex2.cu uses 1 block with 4 threads.

myKernel() code:

```
__global__ void myKernel(int *m, int *v, int *r){
    uint size = 3;
    r[threadIdx.x] = m[0+threadIdx.x*size]*v[0] + m[1+threadIdx.x*size]*v[1] +
m[2+threadIdx.x*size]*v[2];
}
```

The difference between the *lw.cu* and *lw_managed.cu* implementations is how the data is being initialized and stored. In *lw.cu*, the matrix data is present in the CPU memory, and is accessed using pointers. Memory is then allocated in the GPU memory, and the data is copied from the CPU memory to the GPU memory. At this point, the *myKernel* function is called to run on the GPU, updating the final registers in the GPU memory. Finally, the resulting data is copied back into CPU memory from the GPU memory.

Alternatively, in *lw_managed.cu*, the memory allocation uses cudaMallocManaged(), which takes advantage of unified memory. The *myKernel* function is still executed in the GPU kernel, and all the matrix memory is saved on the unified memory. With unified memory, the CPU can access the same matrices in memory that the GPU had written to, so there is no need to copy the memory.

Part 2:

Grayscale only:

Method	**Execution time (ms)**
OpenCV:	**~0.0009** ms per frame
CPU:	**~0.0039** ms per frame
GPU without unified memory:	**~0.0045** ms per frame
GPU with unified memory:	**~0.0028** ms per frame

Grayscale + invert + blur:

Method	**Execution time (ms)**
OpenCV:	**~0.0022** ms per frame
CPU:	**~0.0200** ms per frame
GPU without unified memory:	**~0.0102** ms per frame
GPU with unified memory:	**~0.0108** ms per frame

The OpenCV implementations of the image filtering has the best performance overall. Since the OpenCV functions are well-developed, there are likely to be additional optimizations to the computations separate than using a naïve algorithm with more efficient hardware computations.

Assignment 4 Deliverables

Part 1: Sobel Filter

Runtimes:

- Image size (W x H): 1024 x 1024

Method	**Execution time (ms)**
OpenCV:	**~28.7** ms per frame
CPU:	**~143.0** ms per frame
GPU with unified memory:	**~5.2** ms per frame

- Image size (W x H): 512 x 512

Method	**Execution time (ms)**
OpenCV:	**~7.5** ms per frame
CPU:	**~33.2** ms per frame
GPU with unified memory:	**~2.1** ms per frame

Part 2: Blocked Matrix Multiplication

```
wes-237b@ubuntu:~/Desktop/jupyter/wes237b_lab4/assignment4/matrix$ ./mm 16 16
Time CPU = 0.17ms, Time GPU = -1.00ms, Speedup = -0.17x, RMSE = 0.00000
wes-237b@ubuntu:~/Desktop/jupyter/wes237b_lab4/assignment4/matrix$ ./mm 16 512
Time CPU = 5.20ms, Time GPU = -1.00ms, Speedup = -5.20x, RMSE = 0.00000
wes-237b@ubuntu:~/Desktop/jupyter/wes237b_lab4/assignment4/matrix$ ./mm 512 16
Time CPU = 1.21ms, Time GPU = -1.00ms, Speedup = -1.21x, RMSE = 0.00004
wes-237b@ubuntu:~/Desktop/jupyter/wes237b_lab4/assignment4/matrix$ ./mm 512 512
Time CPU = 204.17ms, Time GPU = -1.00ms, Speedup = -204.17x, RMSE = 0.00004
wes-237b@ubuntu:~/Desktop/jupyter/wes237b_lab4/assignment4/matrix$ ./mm 1024 512
Time CPU = 396.35ms, Time GPU = -1.00ms, Speedup = -396.35x, RMSE = 0.00011
wes-237b@ubuntu:~/Desktop/jupyter/wes237b_lab4/assignment4/matrix$ ./mm 512 1024
Time CPU = 768.39ms, Time GPU = -1.00ms, Speedup = -768.39x, RMSE = 0.00004
wes-237b@ubuntu:~/Desktop/jupyter/wes237b_lab4/assignment4/matrix$ ./mm 1024 1024
Time CPU = 1578.98ms, Time GPU = -1.00ms, Speedup = -1578.98x, RMSE = 0.00011
wes-237b@ubuntu:~/Desktop/jupyter/wes237b_lab4/assignment4/matrix$
```