HW3: Sobel

110065504 葉之晴

1. **How did you parallelize the code?**
    - Which CUDA APIs did you use?

      The API architecture of CUDA is divided into two parts, C language extension and runtime library. For the extension of the C language, I used __device__, __global__, blockIdx, and blockDim. For the other, I used cudaMalloc, cudaMemcpy, cudaFree, and cudaDeviceSynchronize.

    - Which functions are ported to CUDA? How did you distribute the workloal to blocks and threads?

      Since the principle of edge detection is through the convolution of Gx and Gy directions, we need to correctly index the position of each element in the 8*8 small square centered on the target pixel when CUDA is implemented. Because I tried a block from 32*32 , 16*16 and 8*8 and 8*8 is the best. It is a one-dimensional continuous memory that is transmitted from the CPU to the GPU, which increases the difficulty of our indexing. Therefore, in the design of the block and grid, I completely map the entire image to the grid, and each thread corresponds to a pixel. The one-dimensional memory is accurately mapped by the two-dimensional index method.

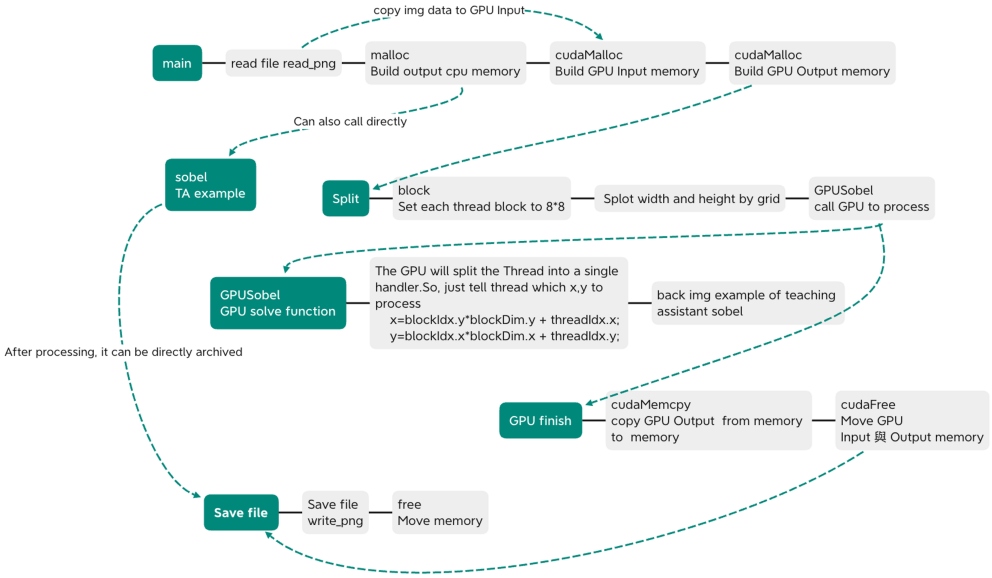
    - How do you implement shared memory?

      Initially, I implemented the mask operations using shared memory in the CUDA environment. However, in these methods for mask operations using shared memory, there are some drawbacks of performance degradation such as increasing the access to the global memory for image replications. So, I didn’t use it in the end

because speed is the priority.

```
__shared__ int mask[MASK_N][MASK_X][MASK_Y];
u = threadIdx.x ;
v = threadIdx.y ;
{
    mask[0][u][v] = dev_mask[0][u][v];
    mask[1][u][v] = dev_mask[1][u][v];
}
```

2. **Which optimization techniques did you apply to your code?**



3. **What's the differenc between cudaMalloc and cudaMallocManaged? When will you pick one over another?**

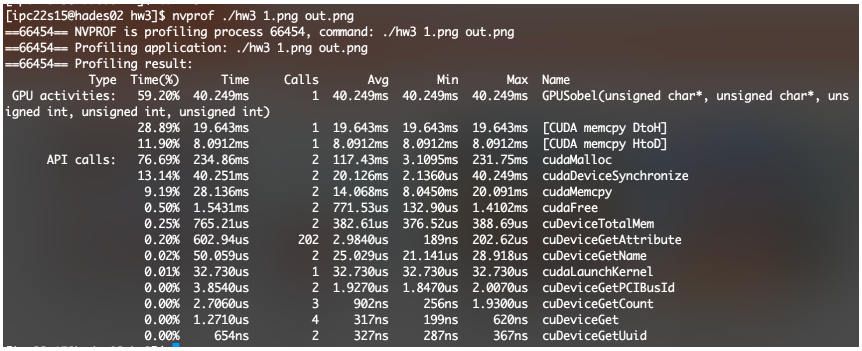
The system moves data from host to device and back, and the code running on the CPU accesses the CPU memory, while the code running on the GPU accesses the GPU memory. We do this with the cudaMallocManaged() function. It works nearly identically to malloc but we have to pass in an address to a pointer as well as the size because cudaMallocManaged returns a cudaError_t instead of a pointer. The example is in the following. In addition, it can be advantageous to manage the memory on the CPU and the GPU in your programs. To accomplish this, you use cudaMalloc() and cudaMemcpy() to allocate and transfer memory. cudaMalloc() is very similar to cudaMallocManaged()

and takes the same arguments. However, the CPU code will not be able to access memory allocated this way.

```
cudaMallocManaged((void**)&ptr, SIZE * sizeof(int))
```

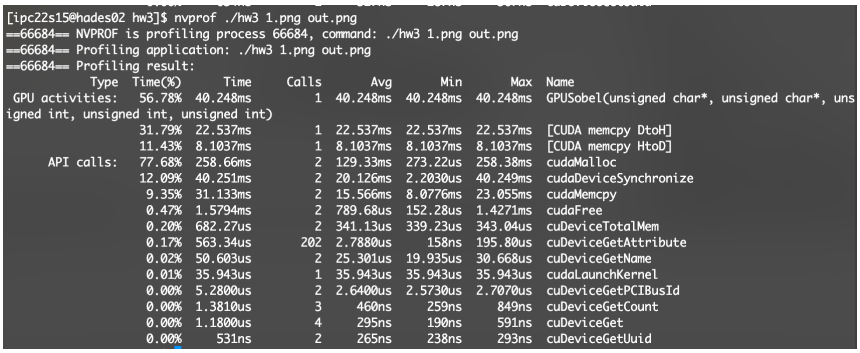
Using cudaMalloc and cudaMemcpy to copy data from memory to GPU memory, saves more time than cudaMallocManaged. So, I choose cudaMalloc.

4. **Experiment:**
    - Measure the GPU kernel time using nvprof. Show the difference with and without shared memory.
    - Profiling your program by measuring the time spend in I/O, memory copy, CPU, and kernel.
        - Without share memory



```
[ipc22s15@hades02 hw3]$ nvprof ./hw3 1.png out.png
==66454== NVPROF is profiling process 66454, command: ./hw3 1.png out.png
==66454== Profiling application: ./hw3 1.png out.png
==66454== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   59.20%  40.249ms         1  40.249ms  40.249ms  40.249ms  GPUSobel(unsigned char*, unsigned char*, uns
igned int, unsigned int, unsigned int)
                  28.89%  19.643ms         1  19.643ms  19.643ms  19.643ms  [CUDA memcpy DtoH]
                  11.90%  8.0912ms         1  8.0912ms  8.0912ms  8.0912ms  [CUDA memcpy HtoD]
      API calls:   76.69%  234.86ms         2  117.43ms  3.1095ms  231.75ms  cudaMalloc
                  13.14%  40.251ms         2  20.126ms  2.1360us  40.249ms  cudaDeviceSynchronize
                   9.19%  28.136ms         2  14.068ms  8.0450ms  20.091ms  cudaMemcpy
                   0.50%  1.5431ms         2  771.53us  132.90us  1.4102ms  cudaFree
                   0.25%  765.21us         2  382.61us  376.52us  388.69us  cuDeviceTotalMem
                   0.20%  602.94us       202  2.9840us     189ns  202.62us  cuDeviceGetAttribute
                   0.02%  50.059us         2  25.029us  21.141us  28.918us  cuDeviceGetName
                   0.01%  32.730us         1  32.730us  32.730us  32.730us  cudaLaunchKernel
                   0.00%  3.8540us         2  1.9270us  1.8470us  2.0070us  cuDeviceGetPCIBusId
                   0.00%  2.7060us         3     902ns     256ns  1.9300us  cuDeviceGetCount
                   0.00%  1.2710us         4     317ns     199ns     620ns  cuDeviceGet
                   0.00%     654ns         2     327ns     287ns     367ns  cuDeviceGetUuid
```

        - With share memory

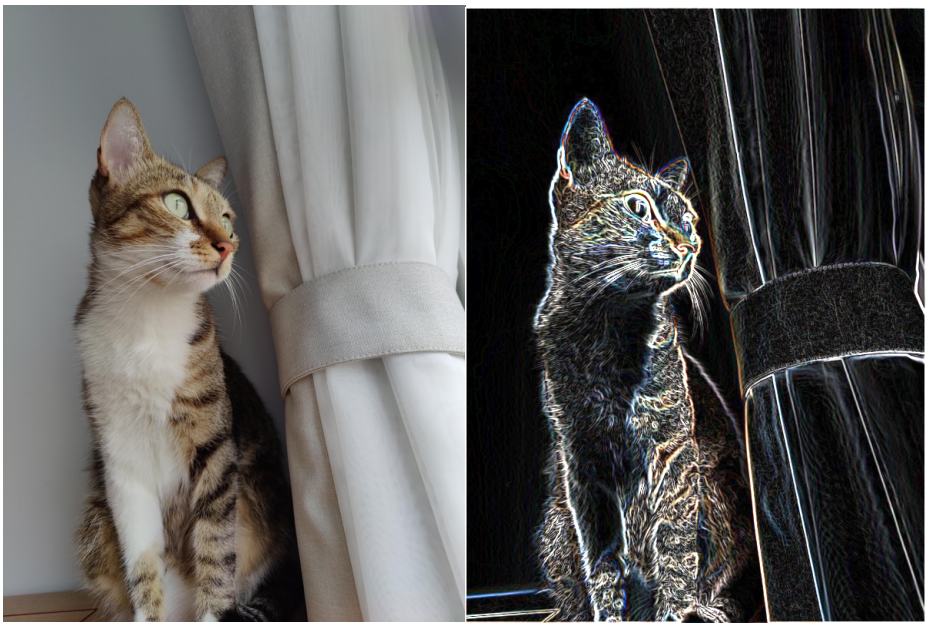


```
[ipc22s15@hades02 hw3]$ nvprof ./hw3 1.png out.png
==66684== NVPROF is profiling process 66684, command: ./hw3 1.png out.png
==66684== Profiling application: ./hw3 1.png out.png
==66684== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   56.78%  40.248ms         1  40.248ms  40.248ms  40.248ms  GPUSobel(unsigned char*, unsigned char*, uns
igned int, unsigned int, unsigned int)
                  31.79%  22.537ms         1  22.537ms  22.537ms  22.537ms  [CUDA memcpy DtoH]
                  11.43%  8.1037ms         1  8.1037ms  8.1037ms  8.1037ms  [CUDA memcpy HtoD]
      API calls:   77.68%  258.66ms         2  129.33ms  273.22us  258.38ms  cudaMalloc
                  12.09%  40.251ms         2  20.126ms  2.2030us  40.249ms  cudaDeviceSynchronize
                   9.35%  31.133ms         2  15.566ms  8.0776ms  23.055ms  cudaMemcpy
                   0.47%  1.5794ms         2  789.68us  152.28us  1.4271ms  cudaFree
                   0.20%  682.27us         2  341.13us  339.23us  343.04us  cuDeviceTotalMem
                   0.17%  563.34us       202  2.7880us     158ns  195.80us  cuDeviceGetAttribute
                   0.02%  50.603us         2  25.301us  19.935us  30.668us  cuDeviceGetName
                   0.01%  35.943us         1  35.943us  35.943us  35.943us  cudaLaunchKernel
                   0.00%  5.2800us         2  2.6400us  2.5730us  2.7070us  cuDeviceGetPCIBusId
                   0.00%  1.3810us         3     460ns     259ns     849ns  cuDeviceGetCount
                   0.00%  1.1800us         4     295ns     190ns     591ns  cuDeviceGet
                   0.00%     531ns         2     265ns     238ns     293ns  cuDeviceGetUuid
```

In my opinion, this is used for multiple sampling theoretically. For example, after 8*8 runs, continue to the next action, or only need to have a high repeatability value and a high probability of repeated operations. However, this program does not use the for

loop, nor does it split processing, so there are not many things in __shared__, and the data is not faster.

5. **Pick any image that is not in the sample test cases, run your implementation with the image, and showcase both the input and output in your report.**



6. **My error experience.**
    - nvprof Error

      RGBA caused nvprof not to execute correctly, so tested for a long time. Then, TA changed the server to me.

    - magick Error

      I haven't been able to get it right with RGBA photos until I remembered to want RGB later. Then magick on the teaching assistant side did not enable sharing correctly, resulting in permission being denied. After the teaching assistant is turned on, it has been converted normally, but the photos I selected do not know where there is a problem, which makes it impossible to convert.

```
[ipc22s26@hades01 ~]$  /home/ipc22/share/hw3/magick mm.png -alpha off dst_1.png.
/home/ipc22/share/hw3/.libs/lt-magick: error while loading shared libraries: libMagickCore-7.Q16HDRI.so.10: car
hared object file: No such file or directory
[ipc22s26@hades01 ~]$ ./magick mm.png -alpha off dst_1.png.
-bash: ./magick: Permission denied
[ipc22s26@hades01 ~]$ ls
hw3  hw3.cu  lab3  magick  Makefile  mm.bmp  mm.out.png  mm.png
[ipc22s26@hades01 ~]$  /home/ipc22/share/hw3/magick ./mm.png -alpha off dst_1.png.
/home/ipc22/share/hw3/.libs/lt-magick: error while loading shared libraries: libMagickCore-7.Q16HDRI.so.10: car
hared object file: No such file or directory
[ipc22s26@hades01 ~]$ magick mm.png -alpha off dst_1.png.
-bash: magick: command not found
[ipc22s26@hades01 ~]$ ./magick ./mm.png -alpha off ./ dst_1.png.
-bash: ./magick: Permission denied
```

- RGBA and RGB Error

  RGBA was not successfully converted to RGB, causing the output to become a black cat. I thought it was funny, so I recorded the mistake this time.