CO316-Assignment 4

Hardik Rana -16CO138
Harshal Shinde -16CO223

Assumption:
For all the questions in the assignment asking "FLOPS per kernel", "Number of reads per kernel" etc it has been assumed that a kernel refers to the operations happening in a single thread.

QUESTION 1 - HISTOGRAMMING NUMBERS

1.Describe all optimizations you tried regardless of whether you committed to them or abandoned them and whether they improved or hurt performance?
Ans: As a optimization technique we tried using shared memory to reduce global memory accesses.

2.Were there any difficulties you had with completing the optimization correctly?
Ans: Yes,It was not possible to store 4096 bins in shared memory while completing the optimization correctly.

3.Which optimizations gave the most benefit?
Ans: Code using **two dimensional thread blocks** gave the maximum benefit in all optimizations.

4.For the histogram kernel, how many global memory reads are being performed by your kernel? explain
Ans: For the histogram kernel there is **1 global memory read** is being performed by our kernel for reading input value.

5.For the histogram kernel, how many global memory writes are being performed by your kernel? explain.
Ans: For the histogram kernel there is **1 write** being performed

by our kernel to increment histogram value,and we have **0 global memory** writes if BIN_CAP is reached.

6.For the histogram kernel, how many atomic operations are being performed by your kernel? explain.

Ans: For the histogram kernel there is **1 atomic add** being performed by our kernel for incrementing histogram value,and we have 0 atomic operations if BIN_CAP is reached.

7.For the histogram kernel, what contentions would you expect if every element in the array has the same value?

Ans: For the histogram kernel if every element in the array has same value then the kernel execution will become very slow due to every thread trying to write in the same location. Hence, it will be akin to sequential execution as atomic operation is done

8.For the histogram kernel, what contentions would you expect if every element in the input array has a random value?

Ans: For the histogram kernel if every element in the array has a random value then the program will run optimally because of the random distribution. Hence, atomic add will cause least number of thread locks.

QUESTION 2 -HISTOGRAMMING ASCII

1.Were there any difficulties you had with completing the optimization correctly?

Ans: Yes,these are the difficulties we had while completing the optimization correctly:

Analysing the problem to realise where **__syncthreads()** should be used and also where **atomicAdd()** to be used was a challenge.

2.Which optimizations gave the most benefit?

Ans: First optimisation was to parallelise the sequential code using global memory. Secondly shared memory was used to create privatized histograms. However transferring this data to global memory was done sequentially. Finally threadIdx.x was used to parallelise transferring privatised histogram to global memory using atomicAdd() . The last optimisation was most beneficial.

3.For the histogram kernel, how many global memory reads are being performed by your kernel? explain

Ans: For the histogram kernel there is **1 global memory read** is performed per thread,and also we have total of **n global memory reads**, where n is the number of ASCII characters.

4.For the histogram kernel, how many global memory writes are being performed by your kernel? explain.

Ans: For the histogram kernel there is **1 global memory write** being performed by our kernel per thread.

5.For the histogram kernel, how many atomic operations are being performed by your kernel? explain.

Ans: For the histogram kernel there are **2 atomic operations** are being performed by our kernel per thread. One atomic operation updates the privatised histogram bins maintained in the shared memory. The second one transfers the shared memory's histogram data to global memory

6.Most text files will consist of only letters, numbers and whitespace characters. These 95 characters make up fall between ASCII numbers 32 - 126. What can we say about atomic access contention if more than 95 threads are simultaneously trying to atomically increment a private histogram?

Ans: The algorithm gives a poor performance in this case. Atomic access contention of privatised histogram bins would yield a result similar to the serial counterpart of the algorithm.

QUESTION 3 -THRUST HISTOGRAM SORT

1.Are there places in your solution where there is an implicit memory copy between the host and device (a copy that is not from thrust::copy)?

Ans: Yes,this the place in our solution where we have an implicit memory copy between the host and device:
When declaring the device vector, the host array pointer is passed as an argument to be copied.

2.What is the asymptotic runtime of this approach to histogram ("bigO" time)?

Ans: The asymptotic runtime of this approach to histogram is
O(n * logn) due to sorting. (n = size of array)

3.What is the asymptotic runtime of the privatized atomic approach to histogram?

Ans: Worst case for atomic approach is when all elements in the array have the same value. Hence, worst case is O(n).
complexity of thrust approach is O(n * logn).

4.Why might the developers of Thrust not provide a straightforward interface?

Ans: The developers of thrust not provide a straightforward interface because doing it serially is probably faster due to the time being wasted to avoid race condition.

QUESTION 4 - CONVOLUTION

1.Name 3 applications of convolution.

Ans: These are the 3 applications of convolution:

- Convolutional neural networks apply multiple cascaded convolution kernels.
- With the help of convolution we can add blurring effect in images
- With the help of convolution we can do edge detection in images

2.How many floating operations are being performed in your convolution kernel? explain.

Ans: Per kernel,2 operations per pixel in the mask (add and multiply). Also, there is a single multiplication being done during clamp. Hence we have total 5 x 5 x 2 + 1 = 51 floating operations in our convolution kernel [5 is mask radius].

3.How many global memory reads are being performed by your kernel? explain.

Ans: we have 1 global memory read per kernel. Where we have 1 to read pixel value and store it in shared memory

4.How many global memory writes are being performed by your kernel? explain.

Ans:we have 0/1 global memory write per kernel.Where we have 1 write to store value after applying mask. 0 if the thread if outside the tile but inside the block.

5.What is the minimum, maximum, and average number of real operations that a thread will perform? Real operations are those that directly contribute to the final output value.

Ans: Floating point operations are real operations here.

The Minimum number of real operations thread will perform is 0.

The Maximum number of real operations thread will perform is 50.

There are **TILE_WIDTH x TILE_WIDTH** threads where 50 operations are done and in (**BLOCK_WIDTH x BLOCK_WIDTH**) - (**TILE_WIDTH xTILE_WIDTH**) kernels, it is zero. Hence, the average is:

(50 x TILE_WIDTH x TILE_WIDTH) / BLOCK_WIDTH x BLOCK_WIDTH.

6.What is the measured floating-point computation rate for the CPU and GPU kernels in this application? How do they each scale with the size of the input?

Ans: Size of Input = M X N

CPU application takes a runtime of **O(M * N**). GPU application has a Parallel

complexity o**f O(1)** with CPU overhead of**(M* N).**

7.How much time is spent as an overhead cost for using the GPU for computation? Consider all code executed within your host function with the exception of the kernel itself, as overhead. How does the overhead scale with the size of the input?

Ans: The overhead is a couple of milliseconds. It increases with input size because of the time taken to read the input matrix and transfer it to the GPU memory. Overhead for transferring back is also higher.

8.What do you think happens as you increase the mask size (say to 1024) while you set the block dimensions to 16x16? What do you end up spending most of your time doing? Does that put other constraints on the way you'd write your algorithm (think of the shared/constant memory size)?

Ans: If mask width is 1024, a couple of megabytes is required to store it. Hence, it

can't be stored in constant memory. This means that every single kernel has to read the mask from global memory. This causes a huge slowdown in kernel execution. You may be forced to read the mask in batches to use the constant memory approach.

9.Do you have to have a separate output memory buffer? Put it in another way, why can't you perform the convolution in place?

Ans: We cannot perform the convolution in place because changing it in place will modify the values of a pixel before it is used in another kernel in another block.

10.What is the identity mask?

Ans: An identity matrix is a square matrix of zeroes except the middle element which is 1.

QUESTION 5 -SHARED MEMORY TILLING

Dimension of matrix: width X height X depth

1.How many global memory reads does your program make?

Ans: Approximately, for every **3 X 3 X 3** tile, **5 X 5 X 5** elements are read from global memory.
Therefore, total global memory reads = **(width * height * depth)/27 * 125**

2.How many shared memory reads does your program make?

Ans: Approximately, every element (except border) would read **7 elements** (including itself) from shared memory.
Therefore, total shared memory reads = **7*(width-2)*(height-2)*(depth-2**)

3.This stencil would need a 3x3 convolution kernel, where the center entry is -6 and the adjacent entries are 1. Corner entries would be 0.

Ans: Not a question

4.Does your stencil make more, equal, or fewer memory accesses than the equivalent 3x3 convolution code would?

Ans: Our code makes fewer global memory accesses than a 3 X 3 convolution code would. However it makes more shared memory accesses