HW1

לידור אלפסי 307854430

דניאל לוי 315668129

Tasks

1.

a. **Our CUDA version is 10.2**
b. **The GPU name is- NVIDIA GeForce RTX 2080 SUPER**
d. **There are 48 multiprocessors in the GPU**

2.

c. **atomicAdd_block() is required for preventing multiple access to the same memory address at the same time. The operation guarantees us that the add operation will be applied by a single thread, without interference from any other thread. Otherwise, the threads can overwrite the other threads operation and cause wrong results.**
The reason we used the block function is because each block works on a different image. So, we need the atomic operation to operate in each block separately.

3.

b. **The data stored in the memory is continuous. Because of that, and because all the threads work on the same data chunks, we can understand that the global memory accesses are coalesced regardless of the number of threads. Each warp accesses 128B data blocks, while its threads work on the same chunk that has been already brought from the global memory.**

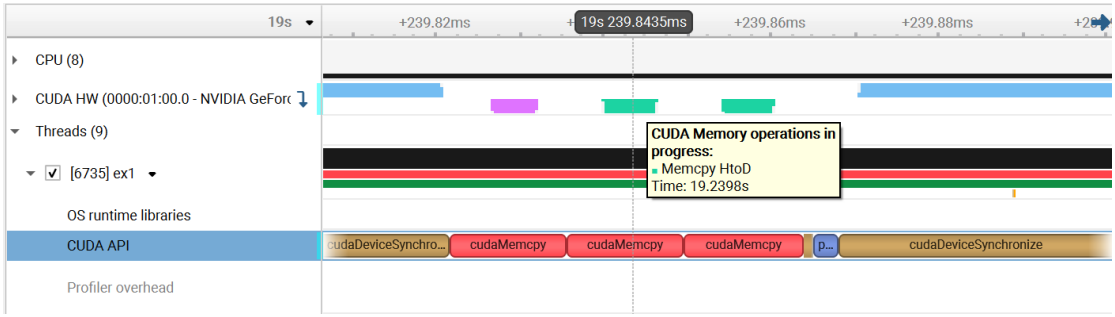
f. **We chose 1024 threads, because we want to maximize the GPU workload while bounded to a single block (for utilizing the shared memory). 1024 is the maximum number of threads each block can run separately.**

g. **The total runtime for the task serial is 9746.441875 [msec]. The throughput is -**

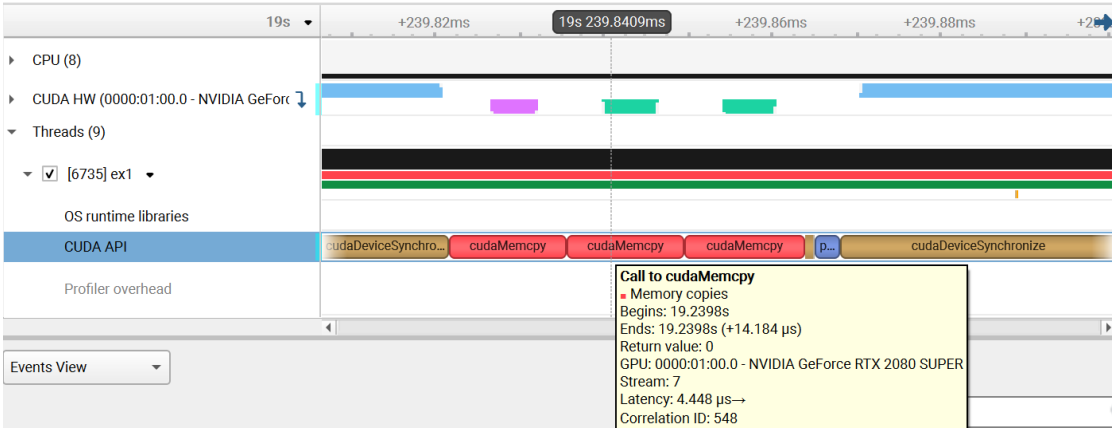
$$Throughput = \frac{images}{runtime} = \frac{10000}{9.746442} = 1026 \frac{images}{sec}$$

h.

i. We found a Memcpy operation from Host to Device



and as reported below, it took $14.184\mu s$

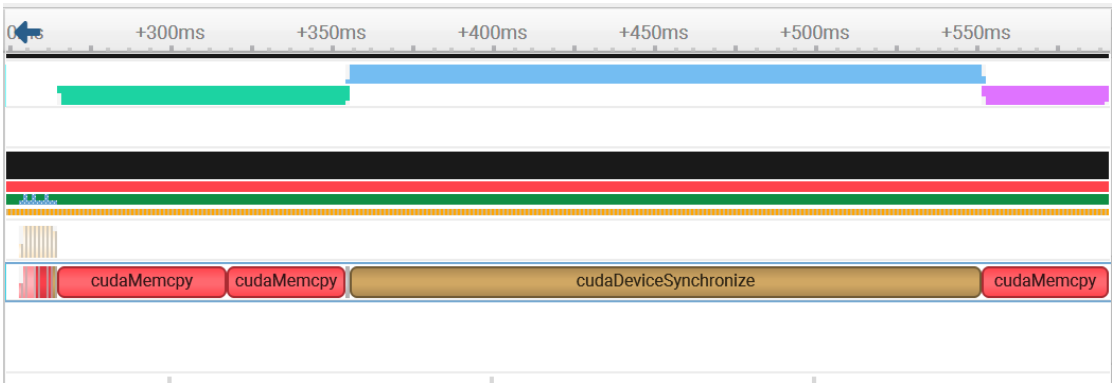


4.

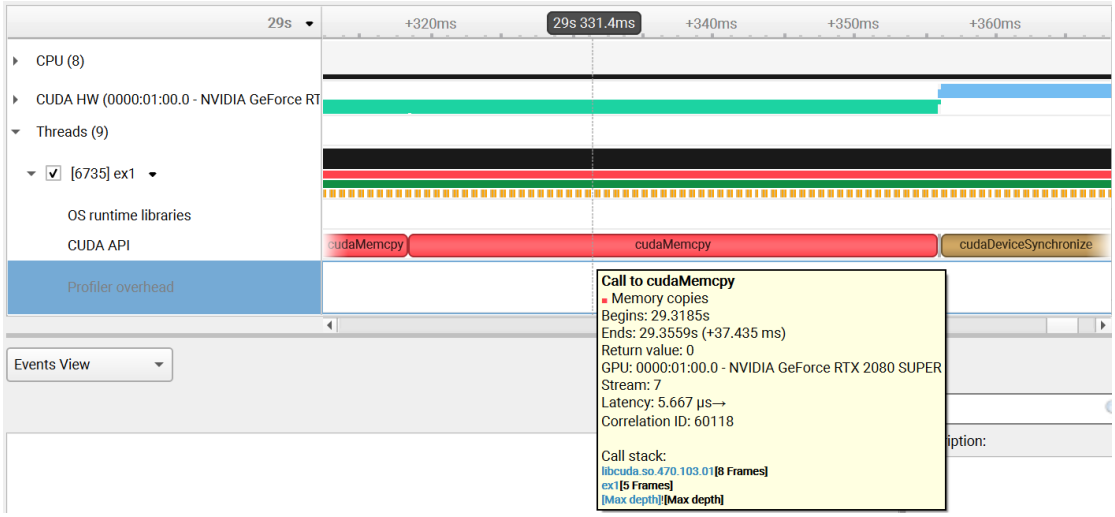
e. The execution time as reported is - 325.354296 [msec]
It means that the speedup we received is -

$$Speedup = \frac{time_{old}}{time_{new}} = \frac{9746.441875}{325.354296} = 29.956$$

f.



g. As we can see-



The Memcpy time is now $37.435ms$.
We copied memory 10000 times bigger than before, in only 2639 times longer latency.
The time doesn't grow linearly with the size of the data being copied.