Deepam Sarmah
2020050
deepam20050@iiitd.ac.in

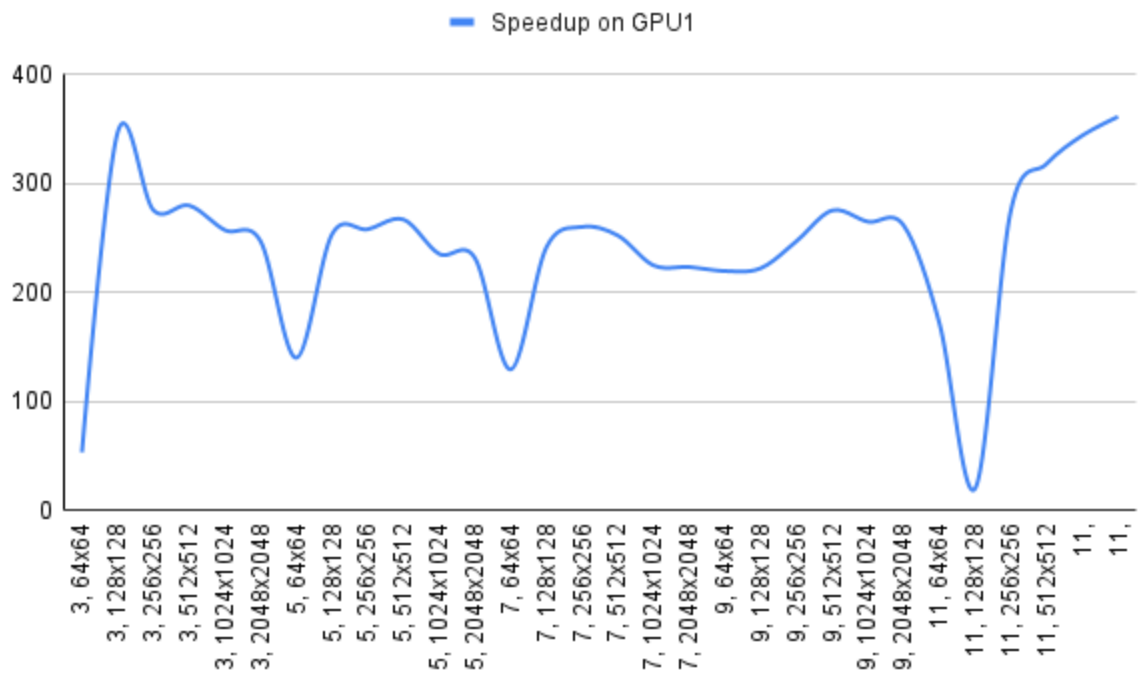
Introduction:
In this assignment I coded up two different GPU implementations, one using the concept of global memory to optimize the given image convolution CPU code and the other using shared memory and constant memory to further optimize the image convolution CPU code.

Strategy:

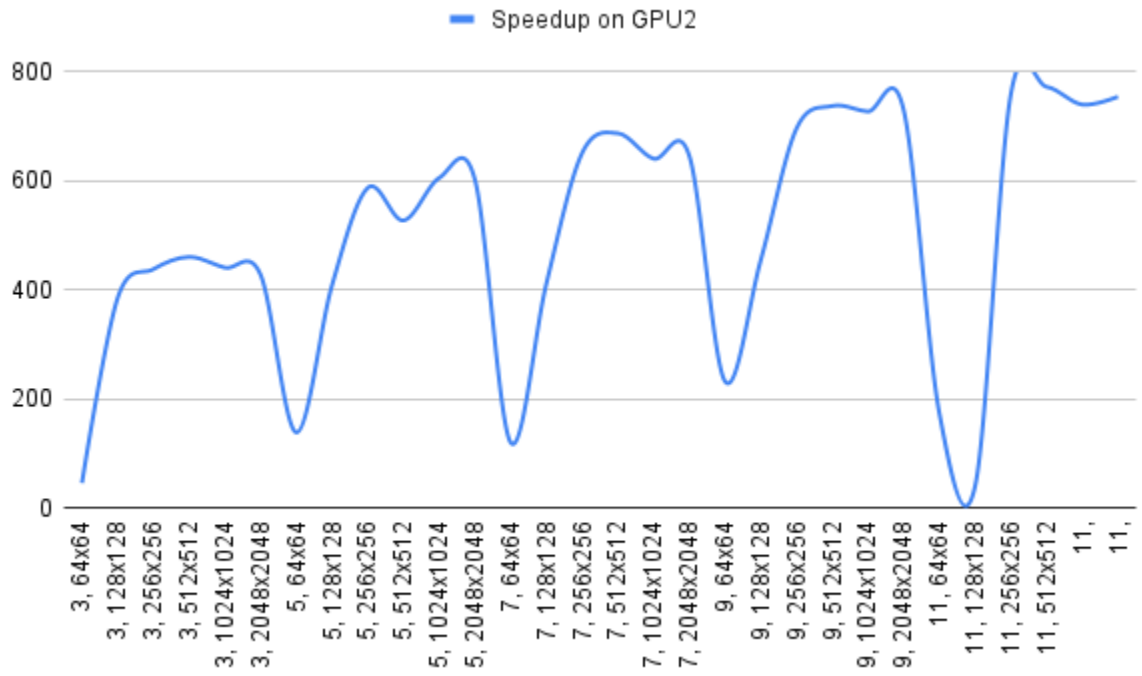
- I created a block consisting of 16x16 threads.
- Each thread of this block would calculate the result for a given (x, y)
- I suitably modified the CPU code to be able to enable each thread to do its own computation from the global memory in the case of GPU1's code.
- In case of GPU2:
    - I created a __shared__ memory array where for each tx, ty (tx and ty being the thread indices in a block in x and y directions respectively) I stored a recurring value that was being used i.e, InputImageData[(yIndex * dataSizeX + xIndex) * channels + k].
    - I noticed that i + m - kCenterY is basically a linear transformation of ty + m and similarly j + n - kCenterX is tx + n. Hence I stored the value of InputImageData[(yIndex * dataSizeX + xIndex) * channels + k] in the shared memory corresponding to the index [ty + m][tx + n]
    - Next I synchronized all the threads in a block.
    - Then whenever I need to use InputImageData[(yIndex * dataSizeX + xIndex) * channels + k] I use the shared memory's value which improves performance.
    - For using constant memory with imageKernel I just define it as __constant__ and make the required changes to copy the data from imageKernel into the constant memory array.
- In the case of using texture memory to read the image for convolution I followed this guide to be able to use the image as texture memory.

Plots:
Y-Axis is the speedup
X-Axis is in format <Kernel, Image Size>

- Speedup on GPU1 vs Kernel, Image Size



- Speedup on GPU2 vs Kernel, Image Size



Results:

Profiling results are in kernel_<i>.txt, i = 3,5,7,9,11

The profiling results also have the results for the bonus question.

All values reported are in microseconds except the speedups.

Kernel	Image Size	CPU	GPU1	GPU2	Speedup on GPU1	Speedup on GPU2	Kernel, Image Size
3	64x64	232.787997	4.384	5.12	53.09945187	45.46640566	3, 64x64
3	128x128	2132.879972	6.144	5.536	347.1484329	385.2745614	3, 128x128
3	256x256	4780.498028	17.344	10.913	275.6283457	438.0553494	3, 256x256
3	512x512	16725.04044	59.744	36.32	279.9451064	460.4912014	3, 512x512
3	1024x1024	60517.14706	235.2	137.28	257.3007953	440.830034	3, 1024x1024
3	2048x2048	229991.9434	930.95	537.44	247.0508012	427.9397577	3, 2048x2048
5	64x64	824.091971	5.888	5.952	139.9612722	138.4563123	5, 64x64
5	128x128	3098.741055	12.224	7.584	253.4964868	408.5892741	5, 128x128
5	256x256	10957.52335	42.464	18.656	258.042656	587.3458054	5, 256x256
5	512x512	42346.86661	158.59	80.289	267.0210392	527.43049	5, 512x512
5	1024x1024	151326.1871	642.85	249.89	235.3989066	605.5711999	5, 1024x1024
5	2048x2048	593923.3398	2564.4	986.18	231.6032366	602.2463849	5, 2048x2048
7	64x64	1262.019992	9.76	10.528	129.305327	119.8727196	7, 64x64
7	128x128	5177.230835	21.44	12.577	241.4753188	411.6427475	7, 128x128
7	256x256	21069.00215	80.96	32.352	260.2396511	651.2426481	7, 256x256
7	512x512	79263.98468	314.11	115.3	252.3446712	687.4586703	7, 512x512
7	1024x1024	284441.803	1264.9	443.91	224.8729567	640.7645761	7, 1024x1024
7	2048x2048	1137704.468	5092.9	1760.9	223.3903017	646.0926048	7, 2048x2048
9	64x64	2636.346102	12	11.36	219.6955085	232.0727202	9, 64x64

9	128x128	7332.498074	32.992	16.128	222.2507903	454.6439778	9, 128x128
9	256x256	31125.30708	125.95	44.704	247.1243119	696.2532902	9, 256x256
9	512x512	121555.5344	441.44	164.8	275.3613953	737.5942619	9, 512x512
9	1024x1024	464861.9385	1754	639.04	265.0296114	727.4379358	9, 1024x1024
9	2048x2048	1853213.867	7084	2537.8	261.6055713	730.2442538	9, 2048x2048
11	64x64	3090.331078	17.984	17.824	171.8378046	173.3803343	11, 64x64
11	128x128	938.66	47.328	23.552	19.83307978	39.8547894	11, 128x128
11	256x256	49602.99683	180.29	65.344	275.1289413	759.1056076	11, 256x256
11	512x512	187166.5649	588.19	242.24	318.2076624	772.6492938	11, 512x512
11	1024x1024	694698.4253	2018.3	938.66	344.1997846	740.0959083	11, 1024x1024
11	2048x2048	2814162.109	7785.5	3729	361.4619625	754.6693777	11, 2048x2048

References:

- https://forums.developer.nvidia.com/t/writing-to-texture-memory/12255/8
- https://github.com/zchee/cuda-sample/blob/master/0_Simple/simpleTexture/simpleTexture.cu