Project5 Write Up

Kernal Implementation

In the project we were given a CPU example of the convolve kernel and were asked to implement GPU version. Writing the GPU kernal was a minor tweak from the CPU version. The two most outer for loop does point-wise loop over the image pixels. I just had to update iterator variable by adding x0= (blockIdx.x * blockDim.x + threadIdx.x) to outer loop that loops thorugh x-axis and y0=(blockIdx.y * blockDim.y + threadIdx.y) to inner loop that loops thorugh y-axis. x0 and y0 gives start of each block thus each block will loop starting from its location.

Striding was implemented by updating the iteration variable of outer for loop by blockDim.x*gridDim.x and updating the iteration variable of inner for loop by adding blocDim.y*gridDim.y which is the total number of thread in the grid. In the case where input cell number is larger than thread available, after blockDim.x*gridDim.x , blocDim.y*gridDim.y, there will still be input-blockDim.x*gridDim.x, and input-blocDim.y*gridDim.y cell to be processed so by striding by blockDim.x*gridDim.x, loop would not terminate and process the left over cells after iteration untill all have been computed

Data distribution

With block size and grid sizes, we distirbute data to thread in each block by starting loop at blockIdx.x * blockDim.x + thradIdx.x. and blockIdx.y*blockDim.y +threadIdx.y in the inner loop.
Since each block have different blockIdx and each thread different threadIdx, each thread in each block will start processing cells at differnt location.

Performance result

Execution times to run the GPU version on the input file video.mp4 with block sizes 2,4,8,16,32
Grid size was adjusted accordingly:
gridSizeX = height/blockDimSize +(1 if X_Limit%blockDimSize != 0, else 0)
gridSizeY = width/blockDimSize +(1 if Y_Limit%blockDimSize != 0, else 0)

where width and height give the dimensions of the video frame.
Video used was 1920 x 1080 width x height

Grid Sizes:

Block dim size 2 => 540 x 960

Block dim size 4 => 270 x 480

Block dim size 8 => 135 x 240

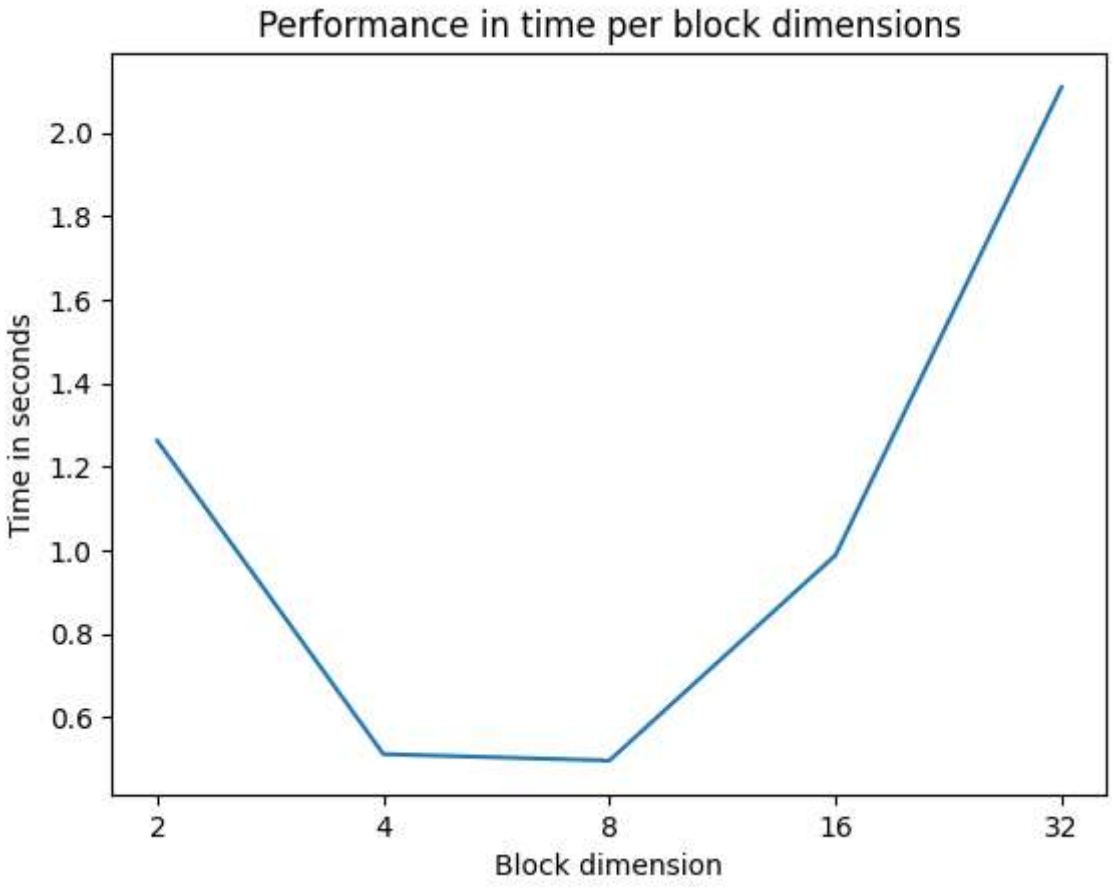
Block dim size 16 => 68 x 120

Block dim size 32 => 34 x 60

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt

x = np.array(['2','4','8','16','32'])
y = np.array([1.26197,0.511747,0.496408,0.987956,2.10882])

plt.plot(x,y)
plt.title("Performance in time per block dimensions")
plt.xlabel("Block dimension")
plt.ylabel("Time in seconds")
plt.show()
```



Performance increased untill block dim size of 8 then gets worse. I did expect that at somepoint when increasing block dim, the peroformance would get worse due to each SM fitting less threadblock resulting in actually less total threads per SM but I did not expect it to happen so soon.