CA Assignment 2

PartB Report

We have tested our code on different sizes of input and kernel matrix and recorded their runtime for basic CPU implementation , basic GPU implementation and optimized GPU implementation.

In optimized GPU implementation , instead of accessing the output array multiple times to update it, we store the result one thread first in the temporary variable and when the computation is completed we store it in the output array.

Here we measure effect of each of your optimizations on the runtime over different input sizes [say, n = 2k, 4k and 8k, We have taken square matrices for both Input Matrix (I) and Output Matrix(O)]

We ran the program Google Colab. The processor specifications are given below:

```
!nvcc --version
```

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
Built on Wed_Sep_21_10:33:58_PDT_2022
Cuda compilation tools, release 11.8, V11.8.89
Build cuda_11.8.r11.8/compiler.31833905_0
```

```
!nvidia-smi
```

```
Fri Nov 24 07:29:12 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.105.17   Driver Version: 525.105.17   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   46C    P8     9W /  70W |      0MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Size of input matrix = 2K *2K

Kernel size:	**16*16**	**64*64**	**256*256**
Reference time(without GPU)	14125ms	202850ms	2.68*10^6ms
Basic implementation (using GPU)	277.153	635.252ms	3584ms
Applying optimization1(temp variable)	141.887ms	310.954ms	1485ms

Size of input matrix = 4K *4K

Kernel Size	**16*16**	**64*64**	**256*256**
Reference time(without GPU)	53563ms	852865ms	
Basic implementation (using GPU)	440ms	1448ms	15010ms
Applying optimization1(temp variable)	252.704ms	691.619	5464ms

Size of input matrix = 8K *8K

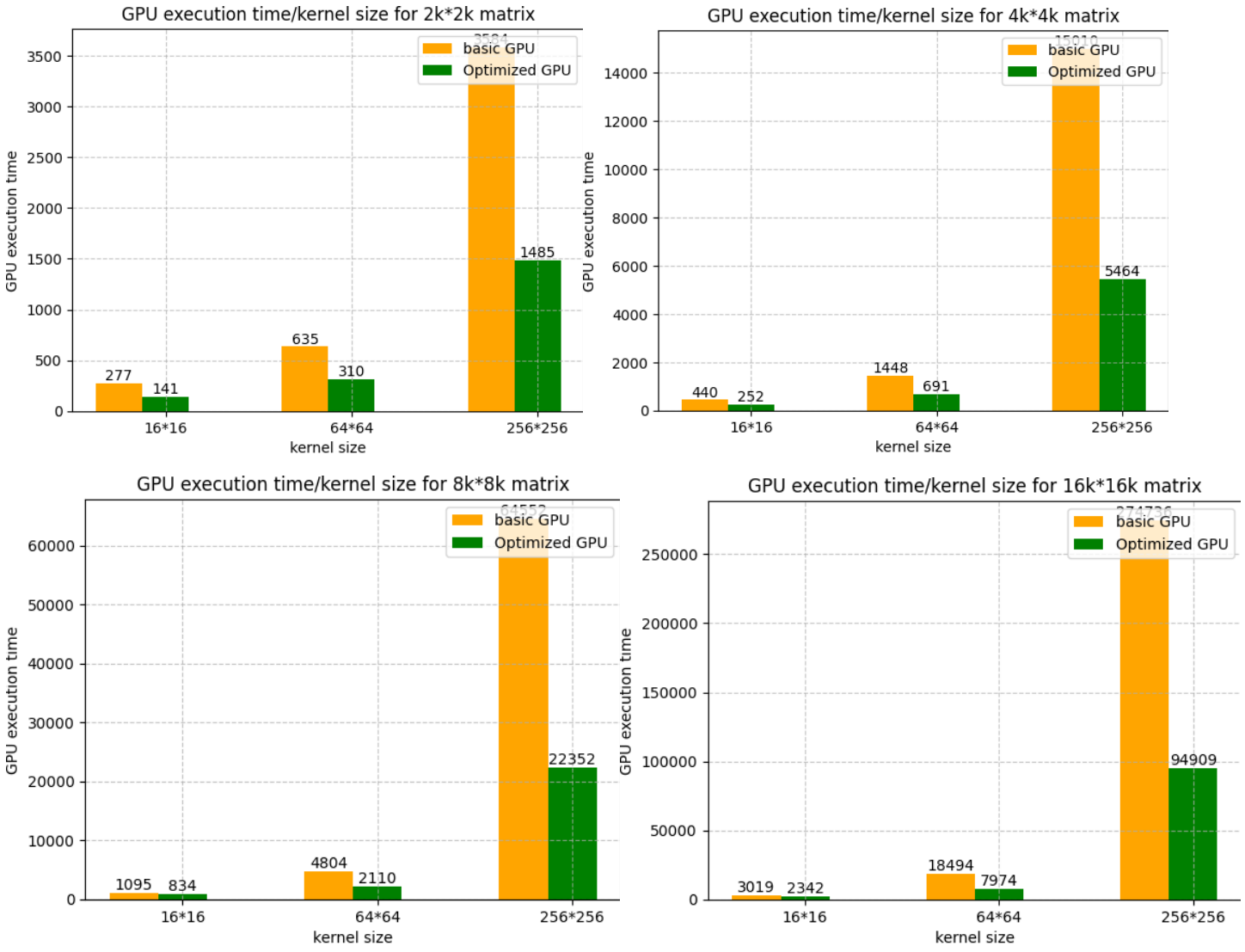
Kernel Size	**16*16**	**64*64**	**256*256**
Reference time(without GPU)	--	3.5*10^7 ms	--
Basic implementation (using GPU)	1095	4804ms	64552ms
Applying optimization1(temp variable)	834ms	2110.29ms	22352ms

Size of input matrix = 16K *16K

Kernel Size	**16*16**	**64*64**	**256*256**
Reference time(without GPU)	--	--	--

Basic implementation (using GPU)	3019ms	18494ms	274736ms
Applying optimization1(temp variable)	2342ms	7974ms	94909ms

Graphs:



NvProf:

Output given by NVProf profiler for input matrix of size 16k*16k and kernel size of 16*16

```
Input file name: data/16384.in
Kernel file name: data/16.in
Input matrix dimensions : 16384x16384
Kernel matrix dimensions : 16x16
==15463== NVPROF is profiling process 15463, command: ./dilated_conv_server
Error: no error
GPU execution time: 2539.63 ms
==15463== Profiling application: ./dilated_conv_server
==15463== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   66.12%  1.42851s         1  1.42851s  1.42851s  1.42851s  [CUDA memcpy DtoH]
                   23.11%  499.22ms         1  499.22ms  499.22ms  499.22ms  convolved(int, int, int*
                   10.77%  232.70ms         2  116.35ms  1.8880us  232.69ms  [CUDA memcpy HtoD]
      API calls:   69.68%  1.66240s         3  554.13ms  90.559us  1.42941s  cudaMemcpy
                   20.93%  499.23ms         1  499.23ms  499.23ms  499.23ms  cudaDeviceSynchronize
                    9.06%  216.22ms         3  72.073ms  116.63us  214.17ms  cudaMalloc
                    0.27%  6.4103ms         3  2.1368ms  942.45us  4.4080ms  cudaFree
                    0.05%  1.2329ms         1  1.2329ms  1.2329ms  1.2329ms  cuDeviceGetPCIBusId
                    0.00%  115.00us       101  1.1380us     135ns  47.418us  cuDeviceGetAttribute
                    0.00%  46.145us         1  46.145us  46.145us  46.145us  cudaLaunchKernel
                    0.00%  24.765us         1  24.765us  24.765us  24.765us  cuDeviceGetName
                    0.00%  2.2070us         3     735ns     237ns  1.5910us  cuDeviceGetCount
                    0.00%  1.0250us         2     512ns     284ns     741ns  cuDeviceGet
                    0.00%     773ns         1     773ns     773ns     773ns  cudaGetLastError
                    0.00%     594ns         1     594ns     594ns     594ns  cuModuleGetLoadingMode
                    0.00%     572ns         1     572ns     572ns     572ns  cudaGetErrorString
                    0.00%     345ns         1     345ns     345ns     345ns  cuDeviceTotalMem
                    0.00%     315ns         1     315ns     315ns     315ns  cuDeviceGetUuid
```

For input_size = 8k*8k and kernel size=16*16:

```
!nvprof ./dilated_conv_server

Input file name: data/8192.in
Kernel file name: data/16.in
Input matrix dimensions : 8192x8192
Kernel matrix dimensions : 16x16
==21153== NVPROF is profiling process 21153, command: ./dilated_conv_server
Error: no error
GPU execution time: 978.903 ms
==21153== Profiling application: ./dilated_conv_server
==21153== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   58.74%  366.93ms         1  366.93ms  366.93ms  366.93ms  [CUDA memcpy DtoH]
                   32.01%  199.96ms         1  199.96ms  199.96ms  199.96ms  convolved(int, int, int*
                    9.25%  57.780ms         2  28.890ms  1.8880us  57.778ms  [CUDA memcpy HtoD]
      API calls:   50.57%  425.84ms         3  141.95ms  86.224us  367.80ms  cudaMemcpy
                   25.18%  212.05ms         3  70.684ms  117.62us  211.31ms  cudaMalloc
                   23.75%  199.98ms         1  199.98ms  199.98ms  199.98ms  cudaDeviceSynchronize
                    0.47%  3.9949ms         3  1.3316ms  415.74us  2.5107ms  cudaFree
                    0.02%  143.69us       101  1.4220us     134ns  61.772us  cuDeviceGetAttribute
                    0.00%  28.279us         1  28.279us  28.279us  28.279us  cudaLaunchKernel
                    0.00%  28.148us         1  28.148us  28.148us  28.148us  cuDeviceGetName
                    0.00%  5.5600us         1  5.5600us  5.5600us  5.5600us  cuDeviceGetPCIBusId
                    0.00%  2.2620us         3     754ns     205ns  1.7110us  cuDeviceGetCount
                    0.00%  1.1360us         2     568ns     290ns     846ns  cuDeviceGet
                    0.00%     872ns         1     872ns     872ns     872ns  cudaGetLastError
                    0.00%     495ns         1     495ns     495ns     495ns  cuModuleGetLoadingMode
                    0.00%     478ns         1     478ns     478ns     478ns  cudaGetErrorString
                    0.00%     429ns         1     429ns     429ns     429ns  cuDeviceTotalMem
                    0.00%     231ns         1     231ns     231ns     231ns  cuDeviceGetUuid
```

When the input size is very large and kernel size is very small , GPU is spending most of the time copying data from device to host.

Conclusion:

1. The GPU implementation of dilated convolution is much faster than the CPU Implementation.
2. The basic implementation of GPU of reduces the computation time by ~98%.
3. Using the temp variable to store partial sum of thread and then taking only single access to update the output array reduces the time significantly, around >50%;
4. This pattern is same for input of larger sizes.
5. Using NVProf Profiler we found that around for larger inputs and very smaller kernel size most of the time is consumed for copying data from GPU to host .