# DD2360
# Assignment 2 - Report

Heng Kang hengkang@kth.se
Jian Wen jianwe@kth.se

**Git repo:** *https://github.com/SinGuLaRiTy2001/DD2360HT23*

## Ex1. First CUDA program and GPU performance metrics

1. Explain how the program is compiled and run.

   **ANSWER:** In Google CoLab, first connect to Google Drive and go to the project directory. Then execute the following commands:

```
!nvcc -arch=sm_75 ./VecAdd.cu -o VecAdd
!./VecAdd 131070
```

2. For a vector length of N:
   1) How many floating operations are being performed in your vector add kernel?

   **ANSWER:** N floating operations are performed.

   2) How many global memory reads are being performed by your kernel?

   **ANSWER:** 2×N global memory reads are performed.

3. For a vector length of 1024:
   1) Explain how many CUDA threads and thread blocks you used.

   **ANSWER:** For a vector with length of 1024, there should be at least 1024 threads. In our program, the threads for each block is 256, and there are (1024 / 256 + 1 = 5) of blocks. The plus one operation is to make sure we have enough threads to handle the load in case of INT format.

   2) Profile your program with Nvidia Nsight. What Achieved Occupancy did you get?

   **ANSWER:** By running the command:

```
!/usr/local/cuda-11/bin/nv-nsight-cu-cli
/content/drive/MyDrive/DD2360/VecAdd 1024 --target-processes all
```

   The Achieved Occupancy is 23.46%. The detailed results are shown below:

```
Section: Occupancy
------------------------------------------------------------------- --------------- ------------------------------
Block Limit SM                                                          block                             16
Block Limit Registers                                                   block                             16
Block Limit Shared Mem                                                  block                             16
Block Limit Warps                                                       block                              4
Theoretical Active Warps per SM                                          warp                             32
Theoretical Occupancy                                                       %                            100
Achieved Occupancy                                                          %                          23.46
Achieved Active Warps Per SM                                             warp                           7.51
------------------------------------------------------------------- --------------- ------------------------------
```

4. Now increase the vector length to 131070:
   1) Did your program still work? If not, what changes did you make?

   **ANSWER:** The program still work. Execution result is given below:

```
✓ 0秒  [72]  !./VecAdd  131070

       The input length is 131070
       Data copy to GPU: 0.888109 ms.
       CUDA kernel: 0.036955 ms.
       Data copy to host: 0.840902 ms.
       Result match reference.
```

   2) Explain how many CUDA threads and thread blocks you used.

   **ANSWER:** As mentioned above, at least 131070 threads are required for such a vector. In our program, there are (131070 / 256 + 1 = 512) blocks, using (512 × 256 = 131072) threads, which is enough for execution.

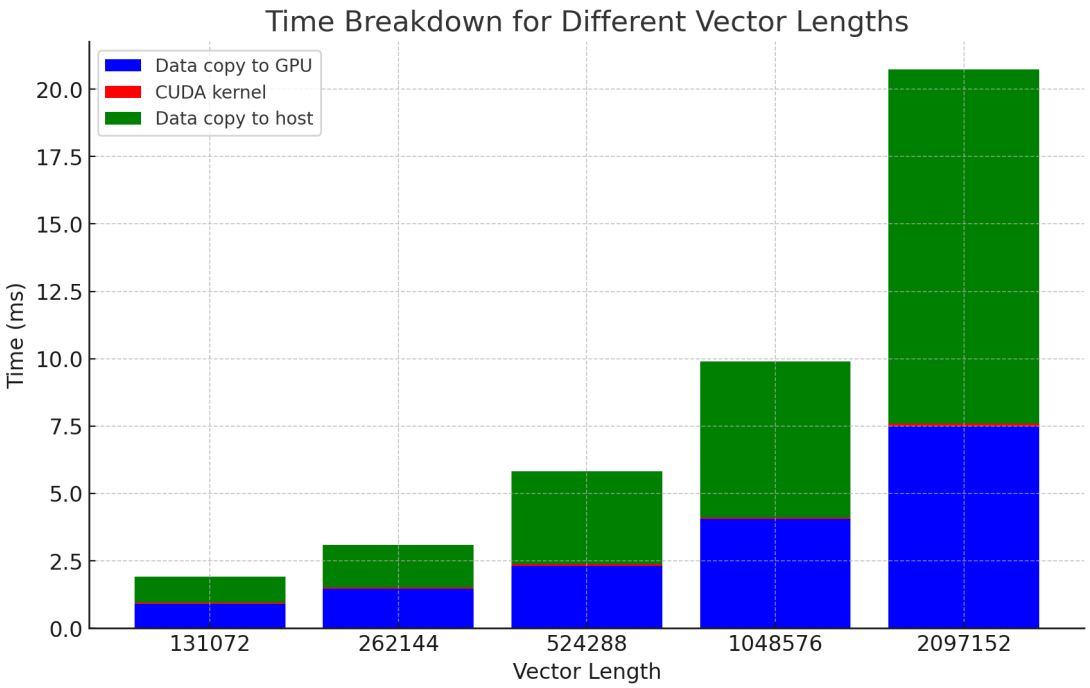
   3) Profile your program with Nvidia Nsight. What Achieved Occupancy did you get?

   **ANSWER:** By using command similar to 3.2, the Achieved Occupancy is 76.55%. The detailed results are shown below:

```
Section: Occupancy
---------------------------------------------------------------------- --------------- ------------------------------
Block Limit SM                                                         block                              16
Block Limit Registers                                                  block                              16
Block Limit Shared Mem                                                 block                              16
Block Limit Warps                                                      block                               4
Theoretical Active Warps per SM                                         warp                              32
Theoretical Occupancy                                                      %                             100
Achieved Occupancy                                                         %                           76.55
Achieved Active Warps Per SM                                            warp                           24.49
---------------------------------------------------------------------- --------------- ------------------------------
```

5. Further increase the vector length (try 6-10 different vector length), plot a stacked bar chart showing the breakdown of time including (1) data copy from host to device (2) the CUDA kernel (3) data copy from device to host. For this, you will need to add simple CPU timers to your code regions.

   **ANSWER:** The breakdown of time is shown in the diagram below:


Time Breakdown for Different Vector Lengths
Data copy to GPU
CUDA kernel
Data copy to host
Time (ms)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
131072
262144
524288
1048576
2097152
Vector Length

## Ex2. 2D Dense Matrix Multiplication

1. Name three applications domains of matrix multiplication.

**ANSWER:** Image Processing, Data Analysis and Engineering simulations.

2. How many floating operations are being performed in your matrix multiply kernel?

**ANSWER:** totalOperations = 2 * numARows * numBColumns * numAColumns;

The number '2' is for both floating add and floating multiply

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

**ANSWER:** totalMemoryReads = 2 * numARows * numBColumns * numAColumns;

The number '2' is for both Matrix A and B need to be read

4. For a matrix A of (128x128) and B of (128x128):
   1. Explain how many CUDA threads and thread blocks you used

      **ANSWER:**128*128 threads will be used, because for each element in the matrix, there will be a thread to take the responsbility. In my code, the number of threads per block in col dimension and number of threads per block in row dimension are 32 , and numbers of blocks in each dimension are 159/32 = 4. Therefore, the number of blocks is 16.
   2. Profile your program with Nvidia Nsight. What **Achieved Occupancy** did you get?

      **ANSWER:** It's 95.85%.

```
Section: Occupancy
------------------------------------------------------------------------------ --------------- ------------------------------
Block Limit SM                                                                        block                             16
Block Limit Registers                                                                 block                              1
Block Limit Shared Mem                                                                block                             16
Block Limit Warps                                                                     block                              1
Theoretical Active Warps per SM                                                        warp                             32
Theoretical Occupancy                                                                     %                            100
Achieved Occupancy                                                                        %                          95.85
Achieved Active Warps Per SM                                                           warp                          30.67
------------------------------------------------------------------------------ --------------- ------------------------------
```

5.For a matrix A of (511x1023) and B of (1023x4094):

1. Did your program still work? If not, what changes did you make?

**ANSWER:** It works.

```
[89]  !./VecMultiply  511 • 1023 • 1023 • 4094

      Input matrix dim (511 x 1023) (1023 x 4094) (511 x 4094)
      The time of data copy from host to device is: 8.290291 ms
      The time of the CUDA Kernel is: 48.376083 ms
      The time of data copy from host to device : 14.031410 ms
      result is correct!
```

2. Explain how many CUDA threads and thread blocks you used.

   **ANSWER:**511 * 4094 threads will be used, because for each element in the matrix, there will be a thread to take the responsbility. In my code, the number of threads per block in col dimension and number of threads per block in row dimension are 32 , and numbers of blocks in row and column dimensions are 16 and 128 respectively. Therefore, the number of thread is 16*128*32*32.

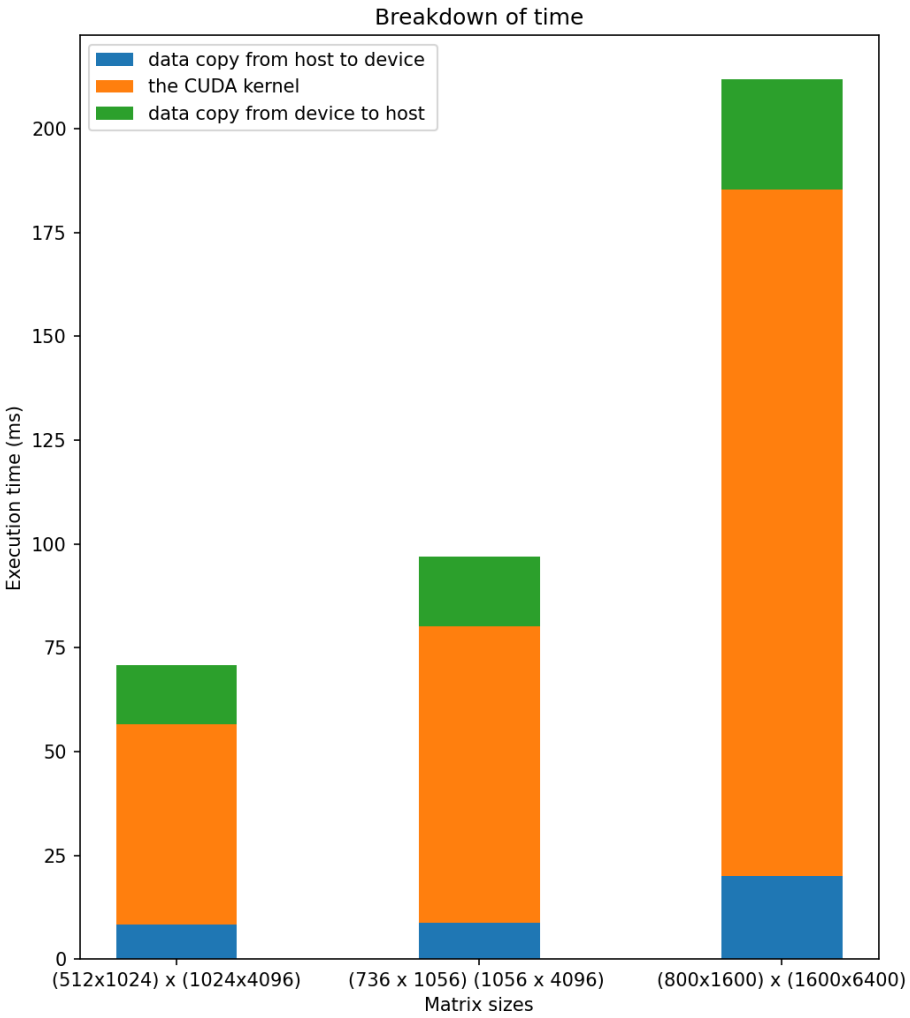
3. Profile your program with Nvidia Nsight. What **Achieved Occupancy** do you get now?

   **ANSWER:** It's 98.01%.

```
Section: Occupancy
---------------------------------------------------------------------- --------------- ------------------------------
Block Limit SM                                                         block                              16
Block Limit Registers                                                  block                               1
Block Limit Shared Mem                                                 block                              16
Block Limit Warps                                                      block                               1
Theoretical Active Warps per SM                                         warp                              32
Theoretical Occupancy                                                      %                             100
Achieved Occupancy                                                         %                           98.01
Achieved Active Warps Per SM                                            warp                           31.36
---------------------------------------------------------------------- --------------- ------------------------------
INF   This kernel's theoretical occupancy is not impacted by any block limit.
```

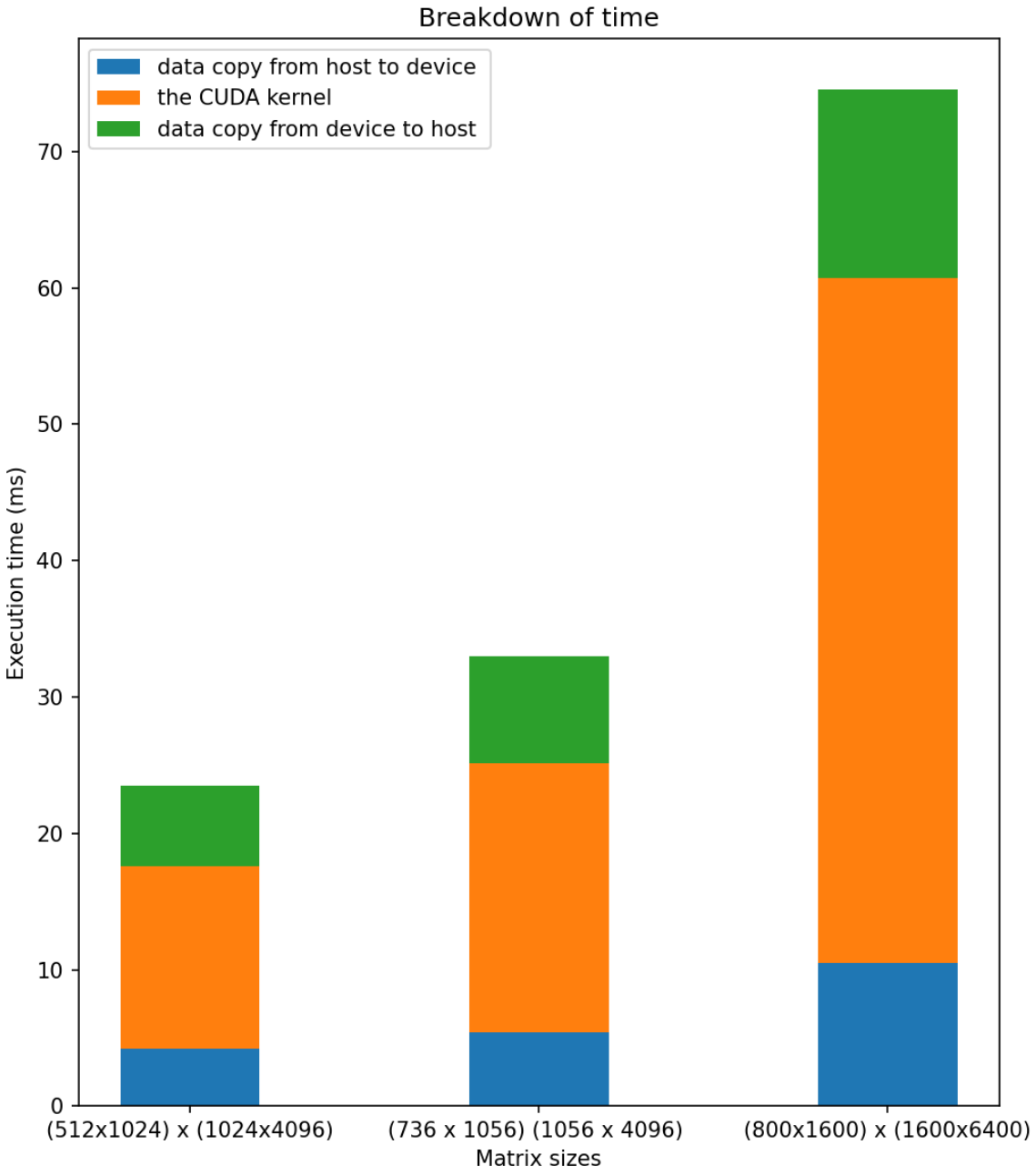
6. Further increase the size of matrix A and B, plot a stacked bar chart showing the breakdown of time including (1) data copy from host to device (2) the CUDA kernel (3) data copy from device to host. For this, you will need to add simple CPU timers to your code regions. Explain what you observe.

**ANSWER:**The time of the CUDA kernel increases significantly, while the other two elements don't change a lot.




7.Now, change DataType from double to float, re-plot the a stacked bar chart showing the time breakdown. Explain what you observe.

**ANSWER:** The time is reduced because double has higher precision than float.




## Contributions

**Jian Wen**

Implemented the exercise 1 (Vector Addition) in the assignment, including cuda program, Nvidia Nsight profile and relative questions.

**Heng Kang**

Implemented the exercise 2 (Vector Multiplication) in the assignment, including cuda program, Nvidia Nsight profile and relative questions.