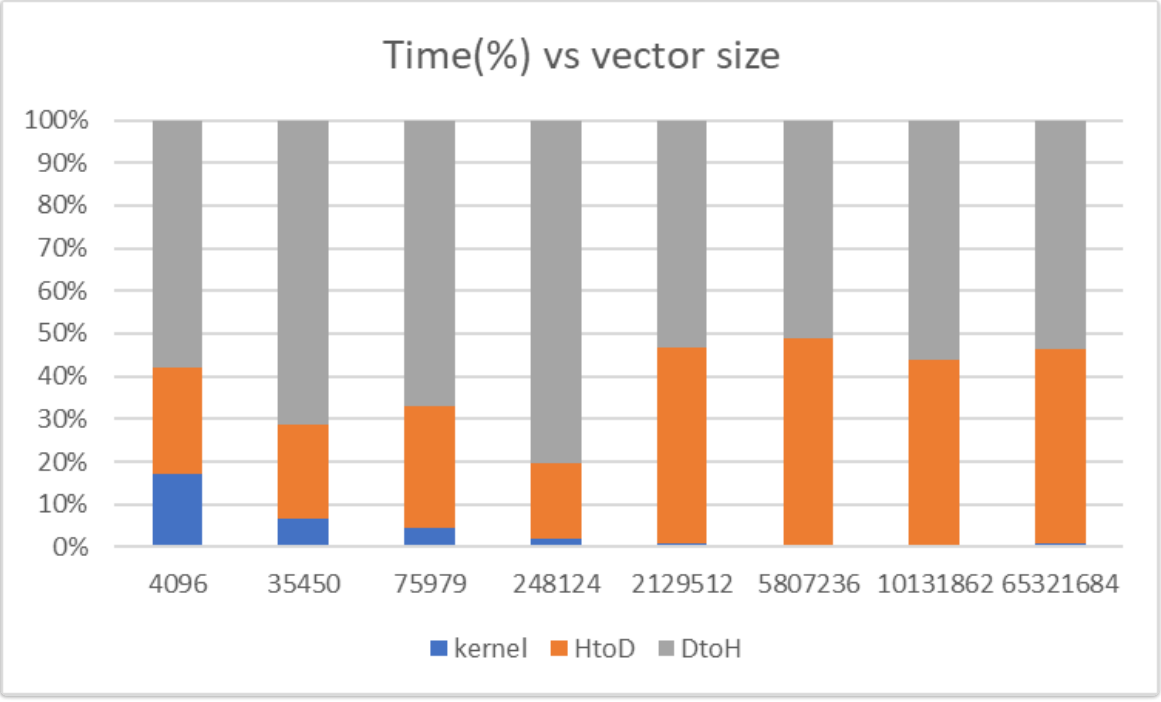
Assignment 2

EXERCISE 1 - YOUR FIRST CUDA PROGRAM AND GPU PERFORMANCE METRICS

1. Explain how the program is compiled and run.
   First, we use `nvcc -o ex1.out ex1.cu` to get the execution file, then use `./ex1.out (vector length) (thread per block)` to assign the vector the in first argument and number of threads per block in second argument.
2. For a vector length of N:
   1. How many floating operations are being performed in your vector add kernel?
      It needs one floating operations when adding two values. So for vector addition for length of N, there is N floating operations.
   2. How many global memory reads are being performed by your kernel?
      It has to read from both vectors, so the global memory reads is $2 \times N$.
3. For a vector length of 1024:
   1. Explain how many CUDA threads and thread blocks you used.
      We set the number of threads per block(TPB) as 512 and the thread block we used is 2. So we use 1024 CUDA threads in this case.
   2. Profile your program with Nvidia Nsight. What Achieved Occupancy did you get?
      Our achieved occupancy is 44.04%
4. Now increase the vector length to 131070:
   1. Did your program still work? If not, what changes did you make?
      Yes, our program still works, so we don't need any change.
   2. Explain how many CUDA threads and thread blocks you used.
      We still set TPB as 512 and this time we have 256 thread blocks, so the CUDA threads are 131072
   3. Profile your program with Nvidia Nsight. What Achieved Occupancy do you get now?
      The achieved occupancy is 76.95%.
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.



EXERCISE 2 - 2D DENSE MATRIX MULTIPLICATION

1. Name three applications domains of matrix multiplication.
   geometric transformation of graphics, back propagation, simulation of time dependent quantum system.

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

   For multiplication of matrices A ($n \times k$) and B ($k \times m$), there are $2 \times n \times k \times m$ floating operations.

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

   For multiplication of matrices A ($n \times k$) and B ($k \times m$), there are $2 \times n \times k \times m$ global memory reads.

4. For a matrix A of (128x128) and B of (128x128):

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

   The threads per block (TPB) is chosen to be 32×32 = 1024, so the blocks per grid (BPG) is calculated as 4×4= 16. In total, we used 16384 threads and 16 thread blocks.

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

   We get 96.99% achieved occupancy.

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

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

      Our program still works, so we make no changes to it.

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

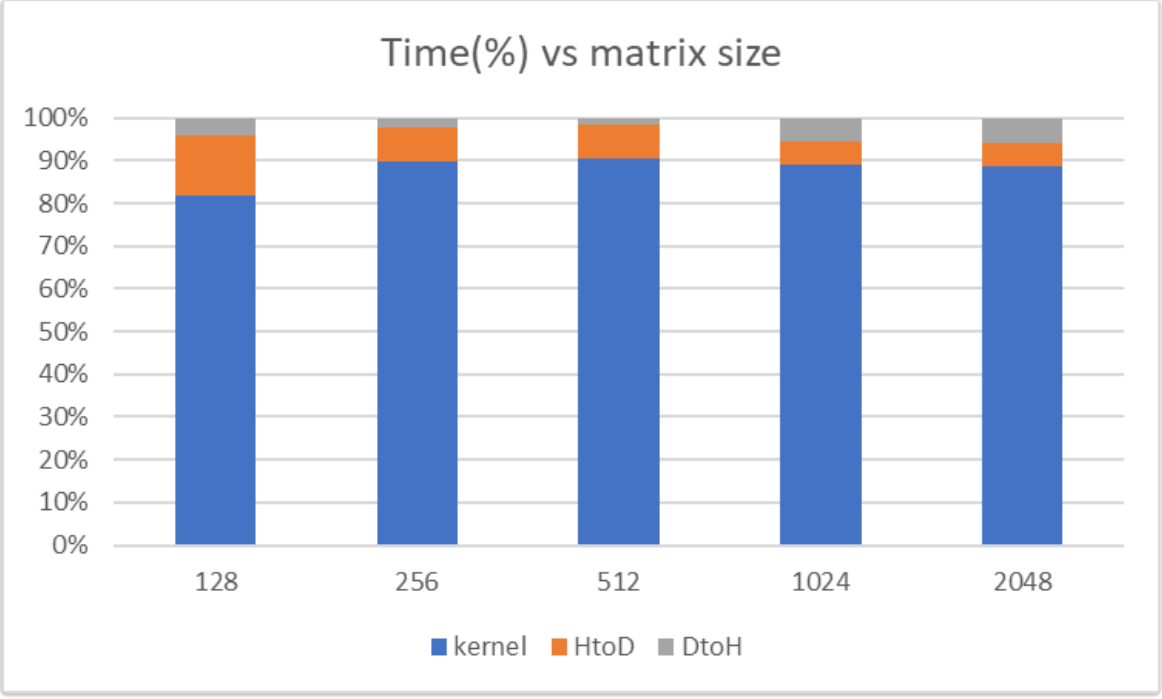
      The threads per block (TPB) is chosen to be 32×32 = 1024, so the blocks per grid (BPG) is calculated as 16×128 = 2048. In total, we used 2097152 threads and 2048 thread blocks.

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

      We get 98.81% achieved occupancy.

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.

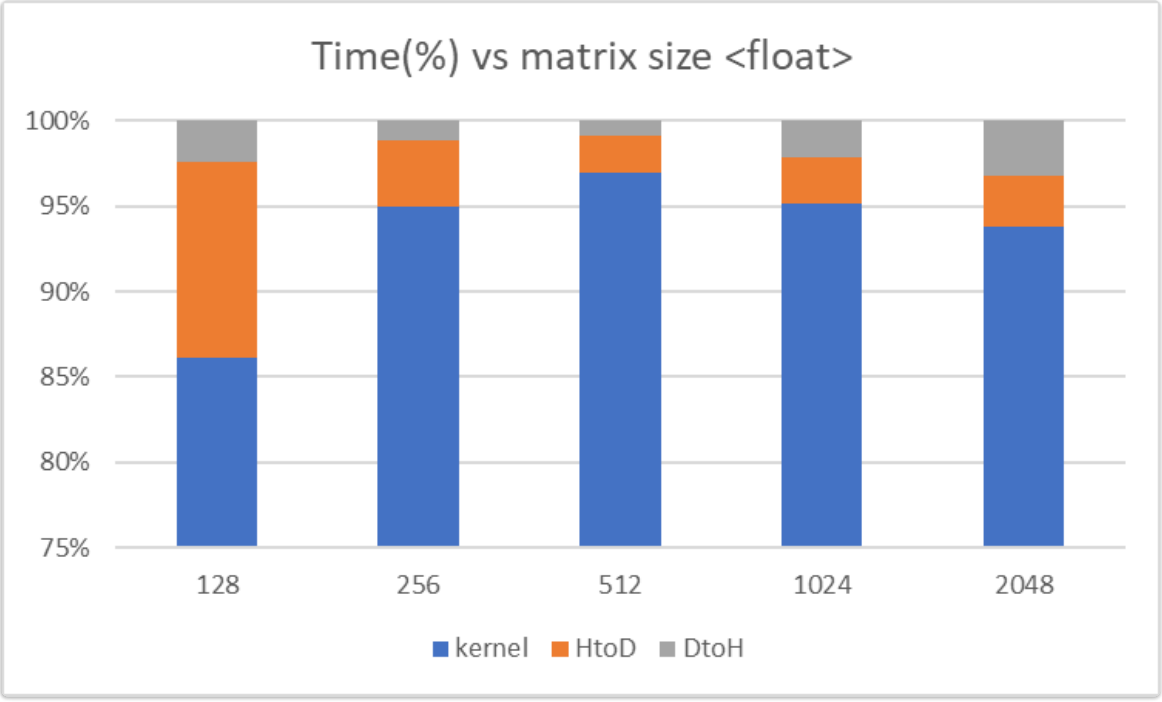
   We used square matrices of different sizes for testing. We can see that the kernel occupies about 90% of the time, and the other 10% is evenly allocated to data copying time.



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

   Both the calculation and the data copying require less time. The allocation of running time is not much different

from the matrix multiplication of doubles.



Contribution: we collaborated to finish both the implementation of code and answering questions.

Github link: https://github.com/LaiYuHong/DD2360HT23/