DD2360 Assignment 2

Group 35

Shiyuan Shan

Github repository: [ShiyuanShan/DD2360HT23 (github.com)](https://github.com/ShiyuanShan/DD2360HT23)

Exercise 1

1.1 The program is compiled with command:

!nvcc -o hw2\_ex1 hw2\_ex1.cu

It is run with command:

!./hw2\_ex1 1024

1.2 1) N floating point addition is done by the kernel.

2) 2N global memory reads are conducted for N adding operations.

1.3 1) Each block has a size of 256. For vector length 1024, 4 thread blocks are needed. Thus in total 4 blocks, 4\*256=1024 threads.

2) Achieved occupancy is 23.44%.

1.4 1) Yes it still works. It works because I have already added the If statement in the kernel so that no overflow takes place.

2) Each block has a size of 256. For vector length 131070, 512 thread blocks are needed. Thus in total 512 blocks, 512\*256=131072 threads.

3) Achieved occupancy is 78.62%.

1.5

Exercise 2

2.1 Matrix multiplication is used in computer graphics, neural network and finite element analysis.

2.2 Assume matrix A has dimension a\_x, a\_y and matrix B has dimension b\_x, b\_y, then a\_y = b\_x. There are a\_x\*b\_y elements in matrix C, each of which is through a\_y multiplication and addition. Thus in total 2\*a\_x\*b\_y\*a\_y floating point operations are performed by the kernel.

2.3 Assume matrix A has dimension a\_x, a\_y and matrix B has dimension b\_x, b\_y, then a\_y = b\_x. There are a\_x\*b\_y elements in matrix C, each of which needs 2\*a\_y global memory reads. Thus in total 2\*a\_x\*b\_y\*a\_y global memory reads are performed by the kernel.

2.4 1) Each block has a dimension of (16,16), 8\*8=64 thread blocks are needed. Thus in total 64 blocks, 64\*16\*16=16384 threads.

2) Achieved occupancy is 42.82%.

2.5 1) Yes it works. The key point should still be the if statement.

2) Each block has a dimension of (16,16), 32\*256=8192 thread blocks are needed. Thus in total 8192 blocks, 8192\*16\*16=2097152 threads.

3) Achieved occupancy is 99.18%.

2.6 Dimension in the following graph means matrix A, B and C are all (Dim, Dim)

Unlike the vector addition case, in which most time is consumed by the memcpy processes, in matrix multiplication the major time consumed is for calculation within the kernel. The relation between time consumed and dimension seems to be a power function.

2.7

The memcpy processes cost only half of the original time. The calculation in kernel consumes are 2-3 times faster than the original time. It is quite sensible because when changed from double to float, memory with same bandwidth can carry twice as much data. Similarly goes for the calculation in kernel, double precision floating point operations are much more time consuming than single precision floating point operations.