CPSC 524 Assignment 5

yl2335, Yangxiaokang Liu

Environment

> [cpsc424_yl2335@c22n05 yl2335_ps5_cpsc424]$ module list
>
> Currently Loaded Modules:
> 1) StdEnv (S) 2) GCCcore/7.3.0 3) binutils/2.30-GCCcore-7.3.0 4) GCC/7.3.0-2.30 5) CUDA/10.1.105 (g)
>
> Where:
> S: Module is Sticky, requires --force to unload or purge
> g: built for GPU

Assumptions

1. Grid dimensions and block dimensions are the same in x and y directions.
2. For task 2, task 3, task 4, tiles and blocks are assumed to have the same size
3. For Task 3 only, the matrix dimensions are assumed to be multiples of the tile width.

Task 1

Part A

The best block and grid dimensions for each run are shown in the table below.

Attributes	Run 1	Run 2	Run 3	Run 4	Run 5
n	1024	8192	1024	8192	8192
m	1024	8192	1024	8192	1024
p	1024	8192	8192	1024	8192
block dim	32	32	32	32	32
grid dim	32	32	32	32	32
gpu time	26.76	13913.49	216.02	1708.31	1792.77
cpu time	272.52	242051.52	2632.11	30737.02	31634.22

Part B

Running time for different block and grid dimensions are shown in the table below.

Attributes	Run 1	Run 2	Run 3	Run 4	Run 5
n	8192	8192	8192	8192	8192
m	8192	8192	8192	8192	8192
p	8192	8192	8192	8192	8192
block dim	2	4	8	16	32
grid dim	4096	2048	1024	512	256
gpu time	307365.06	89548.68	32269.34	19775.64	20339.95

- When using double precision on GPU, the best block dimensions change from (32, 32) to (16, 16) and the grid dimensions change correspondingly from (256, 256) to (512, 512). GPU running time increases from 13913.49 ms to 19775.64 ms

Part C

- Each K80 has 2 GK210 GPUs. Each GK210 GPU has 11 GB, each float takes 4 bytes. Theoretically the maximum matrix size should be sqrt(22GB / 4B / 3) = 44368.
- Each CPU has 6100mb allocated memory, each float takes 4 bytes. Theoretically the maximum matrix size should be sqrt(6GB / 4B / 3) = 23087.
- In practice, the maximum matrix size is 23080. Matrix sizes slightly greater than 23080 can also run successfullly probably due to memory optimization.

  Output:

> n = 23080, m = 23080, p = 23080
>
> Device count = 1
> Using device 0
> Matrix Dimensions = 23080, 23080, 23080
> Block_Dim = (32, 32), Grid_Dim_y = (722, 722),
> Time to calculate results on GPU: 336938.875000 ms.

Task 2

Part A

The best block and grid dimensions for each run are shown in the table below. Note that the best tile width is the same as the best block dim.

Attributes	Run 1	Run 2	Run 3	Run 4	Run 5
n	1024	8192	1024	8192	8192
m	1024	8192	1024	8192	1024
p	1024	8192	8192	1024	8192
block dim	32	32	32	32	32
grid dim	32	32	32	32	32
gpu time	8.02	4205.48	63.85	507.53	648.75
cpu time	283.16	248201.39	2659.02	31236.38	30506.05

- GPU performance is better than that in Task 1 Part A

Part B

Running time for different block and grid dimensions are shown in the table below. Note that the best tile width is the same as the best block dim.

Attributes	Run 1	Run 2	Run 3	Run 4	Run 5
n	8192	8192	8192	8192	8192
m	8192	8192	8192	8192	8192
p	8192	8192	8192	8192	8192
block dim	2	4	8	16	32
grid dim	4096	2048	1024	512	256
gpu time	356022.41	50260.84	9918.34	5968.38	5117.06

- Like Task 1 Part B, switching from single precision to double precision lead to increased GPU time. GPU time increased from 4205.48 ms to 5117.06 ms.
- Unlike Task 1 Part B, switching from single precision to double precision doesn't change the best block dimention nor the best grid dimension. They remain to be (32, 32) and (256, 256).
- GPU performance is better than that in Task 1 Part B.

Task 3

n = p = m = 8192, Block Dims = (32, 32), Grid Dims = (256, 256). Running time for different number of tiles (NTB) are shown in the tables below.

NTB	TIme
2	4384.11
3	4173.80
4	4139.10
5	4135.72
6	4122.19
7	4122.99
8	4124.51
9	4311.28
10	4184.44
11	4167.57
12	4153.05
13	4150.20
14	4151.94
15	4155.88
16	4275.04

- Optimal NTB = 6. Before 6, GPU time decreases, after 6, GPU time flactuates around 4200 ms.
- Using multi-tiled matrix multiplication, GPU time decreases slightly from 4205.48 ms to 4122.19 ms.

Task 4

Output:

> n = 1100, m = 1100, p = 1100, TW = 16, NTB = 3

> Device count = 1
> Using device 0
> Matrix Dimensions = 1100, 1100, 1100
> Block_Dim = (16, 16), Grid_Dim = (23, 69),
> Time to calculate results on GPU: 22.541632 ms.
> Time to calculate results on CPU: 330.359406 ms.
> Scaled error between GPU and CPU: 4.594856e-10

The code works correctly when n = 1100, m = 1100, p = 1100, TW = 16, NTB = 3.