HPML Assignment 5

sss9772@nyu.edu
Shashvat Shah

Part A - Q1

```
vecaddKernel00.o : vecaddKernel00.cu
	${NVCC} $< -c -o $@ $(OPTIONS)

vecadd00 : vecadd.cu vecaddKernel.h vecaddKernel00.o timer.o
	${NVCC} $< vecaddKernel00.o -o $@ $(LIB) timer.o $(OPTIONS)
```

```
Singularity> ./vecadd00 500
Total vector size: 3840000
Time: 0.000364 (sec), GFlopsS: 10.547562, GBytesS: 126.570745
Test PASSED
Singularity> ./vecadd00 1000
Total vector size: 7680000
Time: 0.000765 (sec), GFlopsS: 10.041227, GBytesS: 120.494718
Test PASSED
Singularity> ./vecadd00 2000
Total vector size: 15360000
Time: 0.001518 (sec), GFlopsS: 10.118503, GBytesS: 121.422038
Test PASSED
Singularity>
```

Vector size	Time (sec)	GFlops	GBytes
500	0.000364	10.547562	126.570745
1000	0.000765	10.041227	120.494718
2000	0.001518	10.118503	121.422038

Part A - Q2

```
Singularity> ./vecadd01 500
Total vector size: 3840000
Time: 0.000254 (sec), GFlopsS: 15.108937, GBytesS: 181.307250
3840000 3840000
Test PASSED
Singularity> ./vecadd01 1000
Total vector size: 7680000
Time: 0.000501 (sec), GFlopsS: 15.331868, GBytesS: 183.982416
7680000 7680000
Test PASSED
Singularity> ./vecadd01 2000
Total vector size: 15360000
Time: 0.000993 (sec), GFlopsS: 15.468069, GBytesS: 185.616834
15360000 15360000
Test PASSED
```

Vector size	Time (sec)	GFlops	GBytes	Speedup
500	0.000254	15.108937	181.307250	1.43
1000	0.000501	15.331868	182.982416	1.52
2000	0.000993	15.468069	185.616834	1.53

Part A Q3

```
[sss9772@gr041 PartA]$ ./matmult00 16
number of arguemnts:  2
Data dimensions: 256x256
Grid Dimensions: 16x16
Block Dimensions: 16x16
Footprint Dimensions: 16x16
Time: 0.000038 (sec), nFlops: 33554432, GFlopsS: 879.609302
[sss9772@gr041 PartA]$ ./matmult00 32
number of arguemnts:  2
Data dimensions: 512x512
Grid Dimensions: 32x32
Block Dimensions: 16x16
Footprint Dimensions: 16x16
Time: 0.000216 (sec), nFlops: 268435456, GFlopsS: 1242.715129
[sss9772@gr041 PartA]$ ./matmult00 64
number of arguemnts:  2
Data dimensions: 1024x1024
Grid Dimensions: 64x64
Block Dimensions: 16x16
Footprint Dimensions: 16x16
Time: 0.001590 (sec), nFlops: 2147483648, GFlopsS: 1350.607176
```

DATA DIM	GRID DIM	BLOCK DIM	TIME(SEC)	GFLOPS
256X256	16X16	16X16	0.000038	879.60
512X512	32X32	16X16	0.000216	1242.71
1024X1024	64X64	16X16	0.001590	1350.60

PartA Q4:

```
Singularity> ./matmult01 8
number of arguemnts:  2
Data dimensions: 256x256
Grid Dimensions: 8x8
Block Dimensions: 16x16
Footprint Dimensions: 32x32
Time: 0.000025 (sec), nFlops: 33554432, GFlopsS: 1340.357032
Singularity> ./matmult01 16
number of arguemnts:  2
Data dimensions: 512x512
Grid Dimensions: 16x16
Block Dimensions: 16x16
Footprint Dimensions: 32x32
Time: 0.000119 (sec), nFlops: 268435456, GFlopsS: 2256.312439
Singularity> ./matmult01 32
number of arguemnts:  2
Data dimensions: 1024x1024
Grid Dimensions: 32x32
Block Dimensions: 16x16
Footprint Dimensions: 32x32
Time: 0.000828 (sec), nFlops: 2147483648, GFlopsS: 2593.492443
Singularity>
```

DATA DIM	GRID DIM	BLOCK DIM	TIME(SEC)	GFLOPS	Speedup
256X256	8x8	16X16	0.000025	1340.35	1.52
512X512	16x16	16X16	0.000119	2256.31	1.81
1024X1024	32x32	16X16	0.000828	2593.49	1.92

PartA Q5:

When it comes to optimizing memory access, there are several rules of thumb that can be followed. One effective strategy is to use coalesced reads and writes, which can significantly speed up memory access. By organizing memory reads and writes so that they are accessed in a contiguous block, the GPU can optimize the transfer of data from the device's memory to the processor, resulting in a much faster overall performance.

Another important rule of thumb is to use as many threads as possible. This is because more threads can result in greater parallelism, which can help to break down complex problems into smaller, more manageable parts. By dividing the work into multiple parallel threads, the overall processing time can be greatly reduced, leading to improved performance and faster results.

In addition to using more threads, it is also important to take advantage of shared memory. Shared memory can be used to cache results faster than doing so on the global memory. By using shared memory, multiple threads can access the same data, which can significantly improve overall performance. This is because shared memory is much faster than global memory, allowing threads to access and update data much more quickly. Overall, by following these rules of thumb, developers can greatly improve the performance and efficiency of their GPU-accelerated applications.

PartB Q1

```
Singularity> ./vecAddCpu 1
Time: 0.002380 (sec), GFlopsS: 0.420145, GBytesS: 5.041736
Singularity> ./vecAddCpu 5
Time: 0.012942 (sec), GFlopsS: 0.386344, GBytesS: 4.636127
Singularity> ./vecAddCpu 10
Time: 0.026717 (sec), GFlopsS: 0.374294, GBytesS: 4.491531
Singularity> ./vecAddCpu 50
Time: 0.141983 (sec), GFlopsS: 0.352155, GBytesS: 4.225857
Singularity> ./vecAddCpu 100
Time: 0.271834 (sec), GFlopsS: 0.367871, GBytesS: 4.414457
```

PartB Q2
Blocksize 1

```
Singularity> ./vecAddB2 1
Total vector size: 1000000
Time: 0.098418 (sec), GFlopsS: 0.010161, GBytesS: 0.121929
Test PASSED
Error: 0.000000Singularity> ./vecAddB2 5
Total vector size: 5000000
Time: 0.372792 (sec), GFlopsS: 0.013412, GBytesS: 0.160948
Test PASSED
Error: 0.000000Singularity> ./vecAddB2 10
```

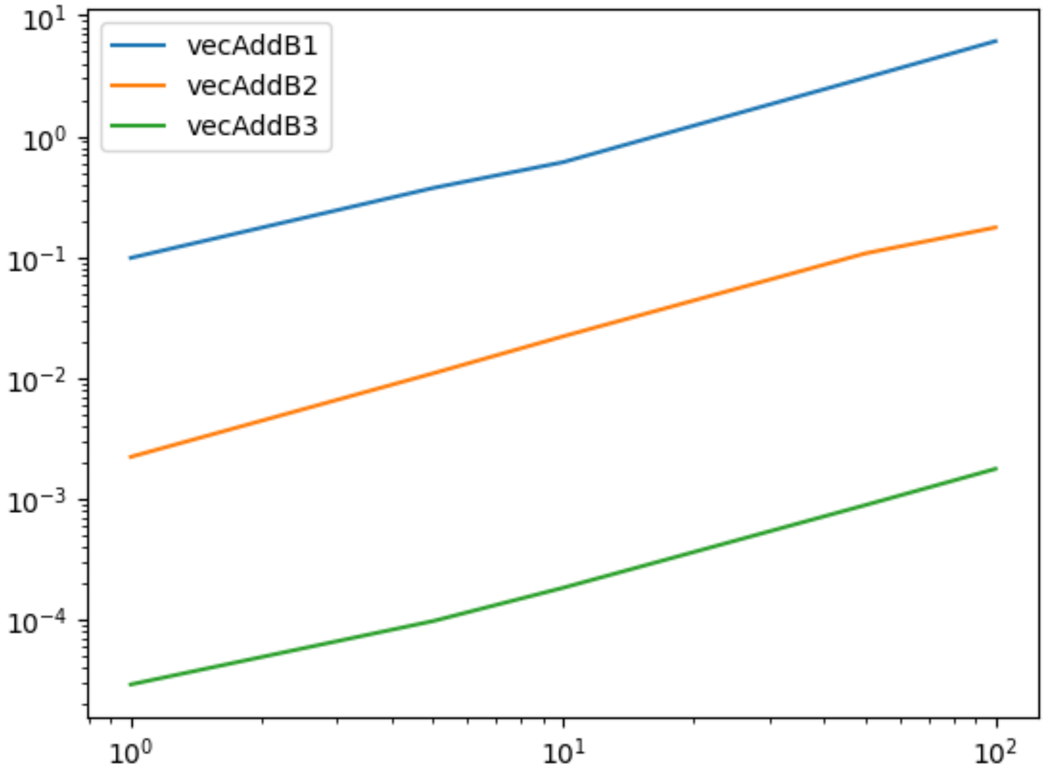
```
Total vector size: 10000000
Time: 0.610343 (sec), GFlopsS: 0.016384, GBytesS: 0.196611
Test PASSED
Error: 0.000000Singularity> ./vecAddB2 50
Total vector size: 50000000
Time: 3.049527 (sec), GFlopsS: 0.016396, GBytesS: 0.196752
Test PASSED
Error: 0.000000Singularity> ./vecAddB2 100
Total vector size: 100000000
Time: 6.128255 (sec), GFlopsS: 0.016318, GBytesS: 0.195814
Test PASSED

Blocksize 256

Singularity> ./vecAddB2 1
Total vector size: 1000000
Time: 0.002219 (sec), GFlopsS: 0.450661, GBytesS: 5.407935
Test PASSED
Error: 0.000000Singularity> ./vecAddB2 5
Total vector size: 5000000
Time: 0.010894 (sec), GFlopsS: 0.458966, GBytesS: 5.507588
Test PASSED
Error: 0.000000Singularity> ./vecAddB2 10
Total vector size: 10000000
Time: 0.022066 (sec), GFlopsS: 0.453184, GBytesS: 5.438202
Test PASSED
Error: 0.000000Singularity> ./vecAddB2 50
Total vector size: 50000000
Time: 0.107247 (sec), GFlopsS: 0.466214, GBytesS: 5.594569
Test PASSED
Error: 0.000000Singularity> ./vecAddB2 100
Total vector size: 100000000
Time: 0.175868 (sec), GFlopsS: 0.568608, GBytesS: 6.823298
Test PASSED

Blocksize 256
GridSize variable

Singularity> ./vecAddB3 1
Total vector size: 1000000
Time: 0.000029 (sec), GFlopsS: 34.379541, GBytesS: 412.554492
Test PASSED
Error: 0.000000Singularity> ./vecAddB3 5
Total vector size: 5000000
Time: 0.000097 (sec), GFlopsS: 51.527076, GBytesS: 618.324914
Test PASSED
Error: 0.000000Singularity> ./vecAddB3 10
Total vector size: 10000000
Time: 0.000183 (sec), GFlopsS: 54.613333, GBytesS: 655.360000
Test PASSED
Error: 0.000000Singularity> ./vecAddB3 50
Total vector size: 50000000
Time: 0.000885 (sec), GFlopsS: 56.496552, GBytesS: 677.958621
Test PASSED
Error: 0.000000Singularity> ./vecAddB3 100
Total vector size: 100000000
Time: 0.001765 (sec), GFlopsS: 56.656815, GBytesS: 679.881778
Test PASSED
```



Simple vector addition

PartB Q3

Blocksize 1 Unified

```
Singularity> ./vecAddB1Unified 1
Time: 0.103535 (sec), GFlopsS: 0.009659, GBytesS: 0.115903
Test PASSED
Error: 0.000000
Singularity> ./vecAddB1Unified 5
Time: 0.383229 (sec), GFlopsS: 0.013047, GBytesS: 0.156564
Test PASSED
Error: 0.000000
Singularity> ./vecAddB1Unified 10
Time: 0.612899 (sec), GFlopsS: 0.016316, GBytesS: 0.195791
Test PASSED
Error: 0.000000
Singularity> ./vecAddB1Unified 50
Time: 3.063598 (sec), GFlopsS: 0.016321, GBytesS: 0.195848
Test PASSED
Error: 0.000000
Singularity> ./vecAddB1Unified 100
Time: 6.127325 (sec), GFlopsS: 0.016320, GBytesS: 0.195844
Test PASSED
Error: 0.000000
```

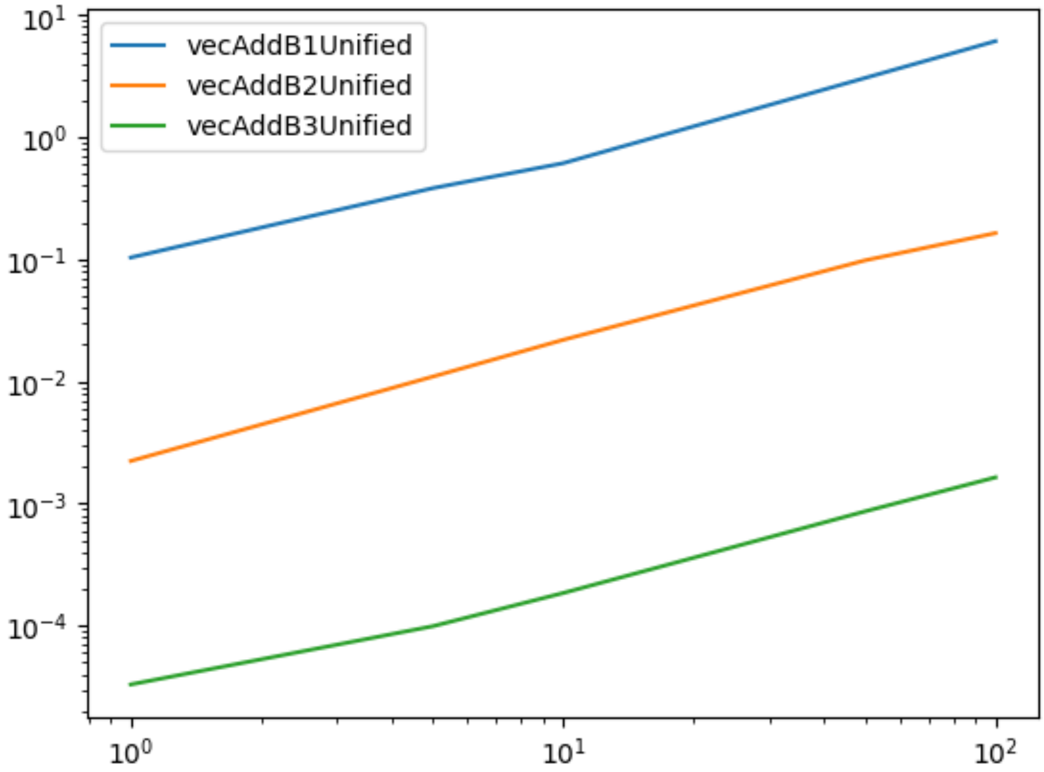
Blocksize 256 Unified

```
Singularity> ./vecAddB2Unified 1
Time: 0.002231 (sec), GFlopsS: 0.448253, GBytesS: 5.379037
```

```
Test PASSED
Error: 0.000000
Singularity> ./vecAddB2Unified 5
Time: 0.010965 (sec), GFlopsS: 0.455992, GBytesS: 5.471902
Test PASSED
Error: 0.000000
Singularity> ./vecAddB2Unified 10
Time: 0.021907 (sec), GFlopsS: 0.456478, GBytesS: 5.477738
Test PASSED
Error: 0.000000
Singularity> ./vecAddB2Unified 50
Time: 0.098250 (sec), GFlopsS: 0.508905, GBytesS: 6.106861
Test PASSED
Error: 0.000000
Singularity> ./vecAddB2Unified 100
Time: 0.164022 (sec), GFlopsS: 0.609674, GBytesS: 7.316082
Test PASSED
Error: 0.000000
Singularity>

Blocksize 256 gridsize variable

Singularity> ./vecAddB3Unified 1
Time: 0.000033 (sec), GFlopsS: 30.393507, GBytesS: 364.722087
Test PASSED
Error: 0.000000
Singularity> ./vecAddB3Unified 5
Time: 0.000099 (sec), GFlopsS: 50.533783, GBytesS: 606.405398
Test PASSED
Error: 0.000000
Singularity> ./vecAddB3Unified 10
Time: 0.000185 (sec), GFlopsS: 54.050309, GBytesS: 648.603711
Test PASSED
Error: 0.000000
Singularity> ./vecAddB3Unified 50
Time: 0.000864 (sec), GFlopsS: 57.868433, GBytesS: 694.421192
Test PASSED
Error: 0.000000
Singularity> ./vecAddB3Unified 100
Time: 0.001639 (sec), GFlopsS: 61.016933, GBytesS: 732.203200
Test PASSED
Error: 0.000000
```



Vector addition with unified memory access

PartC Q1

```
Singularity> ./conv
Time: 0.0041 (ms)
C1_checksum = 122756344698240.000000
```

PartC Q2

```
Singularity> ./conv_tiled
Time: 0.003099 (ms)
C2_checksum = 122756344698240.000000
```

PartC Q3

```
Singularity> ./conv_cudnn
Time: 0.012875 (ms)
C3_Checksum: 122756344698240.000000
```

	Checksum	Time (ms)
C1	122756344698240.000000	0.0041
C2	122756344698240.000000	0.0031
C3	122756344698240.000000	0.0128

References

1. https://siboehm.com/articles/22/CUDA-MMM

2. https://developer.nvidia.com/blog/how-access-global-memory-efficiently-cuda-c-kernels/
3. https://leimao.github.io/blog/CUDA-Coalesced-Memory-Access/