CPEG655

Lab 04

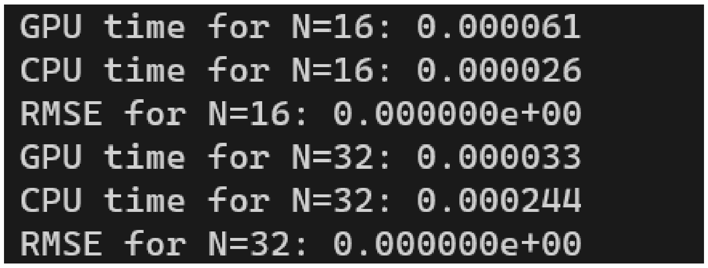
Michael Hutts

Problem 1: NVIDIA CUDA

Task A:

Approach

In this part, a basic implementation of matrix multiplication on GPU was implemented. It performed well and had showed no difference in results compared to the CPU implementation according to RMSE. The RMSE function was also verified for correctness by arbitrarily modifying the values of the final matrices, which did show an extremely high value, showing our metric of correctness was performing correctly and not just giving false confidence.

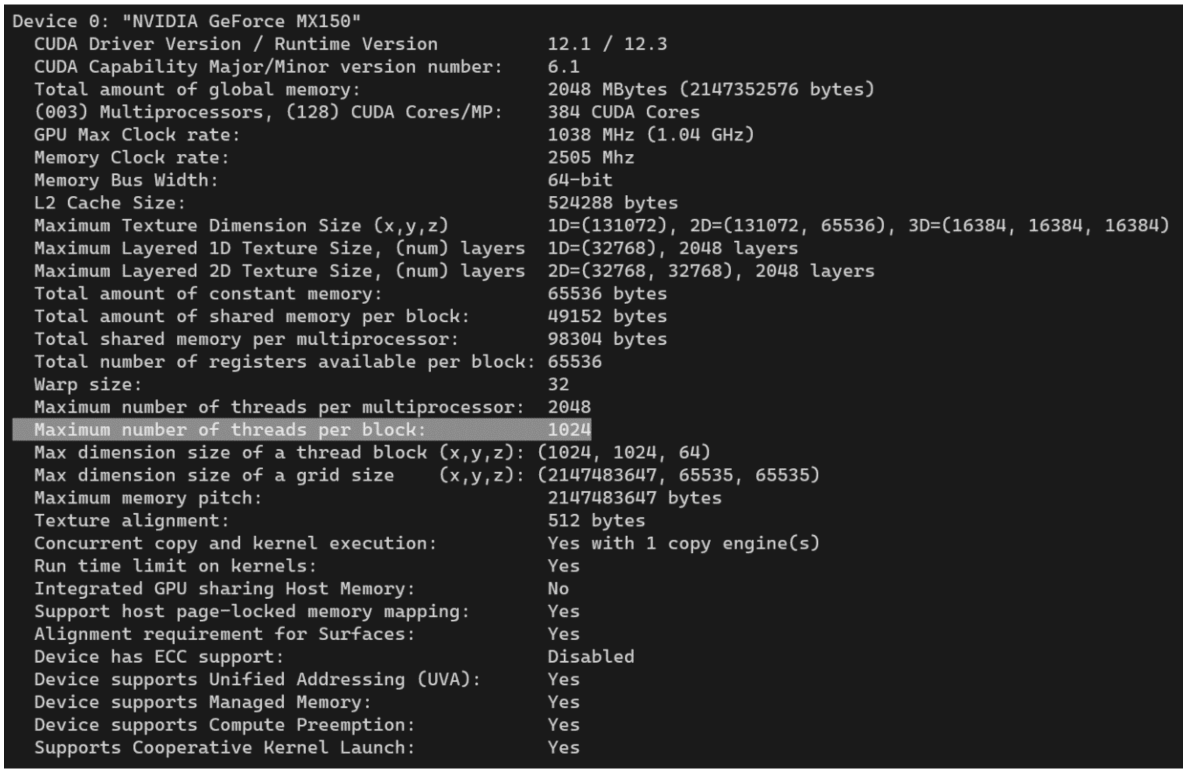


```
GPU time for N=16: 0.000061
CPU time for N=16: 0.000026
RMSE for N=16: 0.000000e+00
GPU time for N=32: 0.000033
CPU time for N=32: 0.000244
RMSE for N=32: 0.000000e+00
```

Figure 1: Example Run

A problem I encountered that I was unable so solve was with cudaEvent_t. Despite it being shown as simple to implement in the slides, for reasons I can't seem to understand using it caused cascading issues in my entire codebase. For this reason I simply used gettimeofday() in place of CUDA events. (Note: this was later fixed for part C as it disregards CPU time, but I chose to leave it out of A and B regardless as using a different timing method between the CPU and GPU was giving bad data)

One additional decision made in this part was the initial value for threadsPerBlock(), although for part A we're limited to 1 block, the number of threads per block is unrestricted. Therefore based on my system information shown below I chose (32, 32) based on the assumption that this should maximize the capability of my GPU for any following workload.

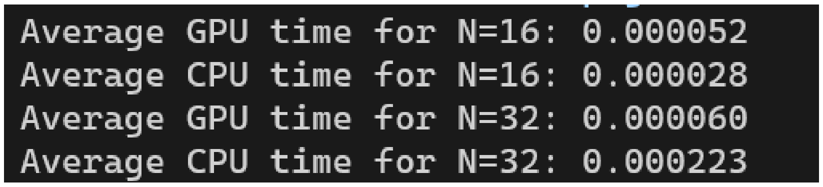


```
Device 0: "NVIDIA GeForce MX150"
  CUDA Driver Version / Runtime Version          12.1 / 12.3
  CUDA Capability Major/Minor version number:    6.1
  Total amount of global memory:                 2048 MBytes (2147352576 bytes)
  (003) Multiprocessors, (128) CUDA Cores/MP:    384 CUDA Cores
  GPU Max Clock rate:                            1038 MHz (1.04 GHz)
  Memory Clock rate:                             2505 Mhz
  Memory Bus Width:                              64-bit
  L2 Cache Size:                                 524288 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total shared memory per multiprocessor:        98304 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 1 copy engine(s)
  Run time limit on kernels:                     Yes
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Managed Memory:                Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
```

Figure 3: GPU Information

Performance

The performance characteristics of part A showed a substantial performance difference for the two sizes of N. For a size of 8 the CPU held a small lead, most likely due to the additional overhead and complexity of a GPU implementation of such a short task, meanwhile for an N of 16 the GPU implementation pulled ahead dramatically, showing the extreme uplift CUDA provides with larger (wider) workloads that can take better advantage of concurrency.

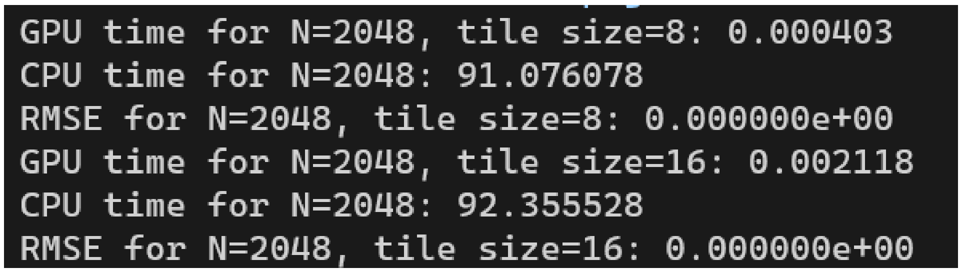


```
Average GPU time for N=16: 0.000052
Average CPU time for N=16: 0.000028
Average GPU time for N=32: 0.000060
Average CPU time for N=32: 0.000223
```

Figure 4: Performance Results

Task B:

For part B, we were no longer limited to just one thread block. This provided substantial opportunity for better performance, but with that required a bit more complexity in making the implantation threadsafe. Additionally, shared memory became a necessity to handle. Despite the additions, the basic structured remained from part A. Additionally the kernel launching process was updated accordingly. With these improvements to concurrency and data locality, an even more extreme performance uplift was measured on a large value of N.

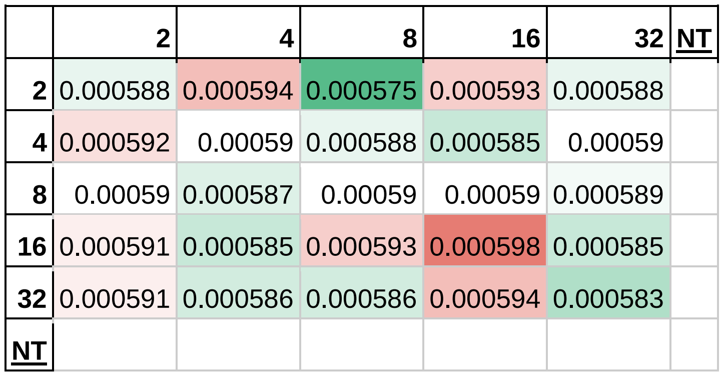


```
GPU time for N=2048, tile size=8: 0.000403
CPU time for N=2048: 91.076078
RMSE for N=2048, tile size=8: 0.000000e+00
GPU time for N=2048, tile size=16: 0.002118
CPU time for N=2048: 92.355528
RMSE for N=2048, tile size=16: 0.000000e+00
```

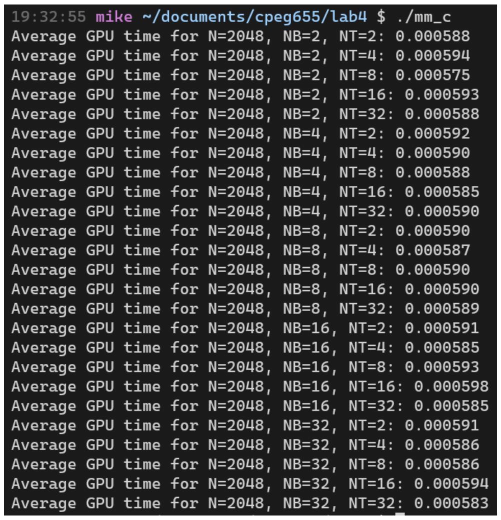
Figure 5: GPU vs CPU Performance Comparison

Part C

For part C, the implementation was relatively straightforward, basically just reworking main() to have modular values NB, NT, and NK as described. Disappointingly the results were not very informative. Although some basic patterns emerged, I could not seem to get consistent enough results to draw a real conclusion on the ideal values of NB and NT. My assumption was that there should be a direct correlation between higher values for both NB and NT and better performance, but not much of a difference seemed to appear.



	2	4	8	16	32	**NT**
2	0.000588	0.000594	0.000575	0.000593	0.000588	
4	0.000592	0.00059	0.000588	0.000585	0.00059	
8	0.00059	0.000587	0.00059	0.00059	0.000589	
16	0.000591	0.000585	0.000593	0.000598	0.000585	
32	0.000591	0.000586	0.000586	0.000594	0.000583	
NT						

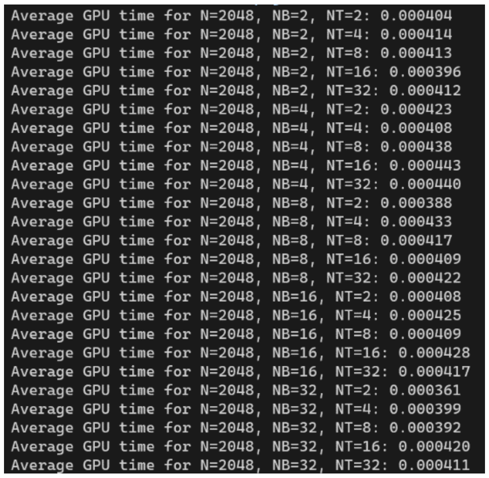


```
19:32:55 mike ~/documents/cpeg655/lab4 $ ./mm_c
Average GPU time for N=2048, NB=2, NT=2: 0.000588
Average GPU time for N=2048, NB=2, NT=4: 0.000594
Average GPU time for N=2048, NB=2, NT=8: 0.000575
Average GPU time for N=2048, NB=2, NT=16: 0.000593
Average GPU time for N=2048, NB=2, NT=32: 0.000588
Average GPU time for N=2048, NB=4, NT=2: 0.000592
Average GPU time for N=2048, NB=4, NT=4: 0.000590
Average GPU time for N=2048, NB=4, NT=8: 0.000588
Average GPU time for N=2048, NB=4, NT=16: 0.000585
Average GPU time for N=2048, NB=4, NT=32: 0.000590
Average GPU time for N=2048, NB=8, NT=2: 0.000590
Average GPU time for N=2048, NB=8, NT=4: 0.000587
Average GPU time for N=2048, NB=8, NT=8: 0.000590
Average GPU time for N=2048, NB=8, NT=16: 0.000590
Average GPU time for N=2048, NB=8, NT=32: 0.000589
Average GPU time for N=2048, NB=16, NT=2: 0.000591
Average GPU time for N=2048, NB=16, NT=4: 0.000585
Average GPU time for N=2048, NB=16, NT=8: 0.000593
Average GPU time for N=2048, NB=16, NT=16: 0.000598
Average GPU time for N=2048, NB=16, NT=32: 0.000585
Average GPU time for N=2048, NB=32, NT=2: 0.000591
Average GPU time for N=2048, NB=32, NT=4: 0.000586
Average GPU time for N=2048, NB=32, NT=8: 0.000586
Average GPU time for N=2048, NB=32, NT=16: 0.000594
Average GPU time for N=2048, NB=32, NT=32: 0.000583
```

Figures 6 & 7: NB/NT Benchmark (1000 Runs per Value)

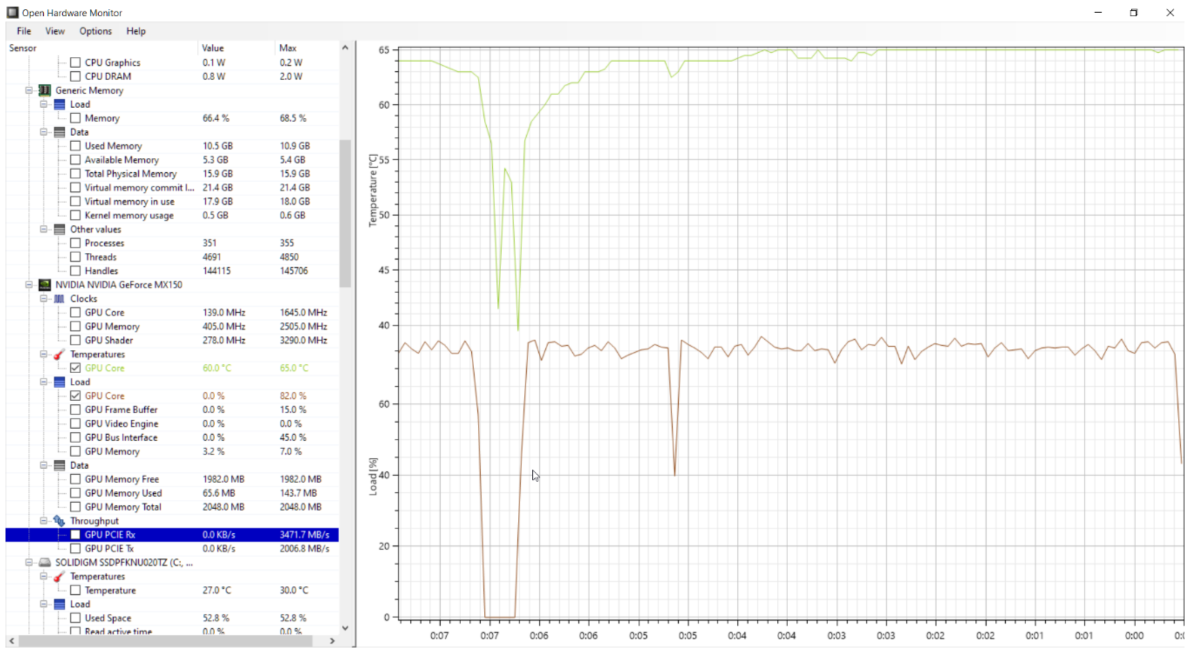
With a small sample size, more dramatic performance differences were shown, but with many repeated attempts thermal throttling would hit the GPU extremely fast. This seems to be a side-effect of the weak cooler on the GPU I'm using. At below 65 degrees celcius the performance would be much better, but hard to benchmark as within a few runs the weak laptop cooler hit this limit, greatly decreasing performance.

	2	4	8	16	32	NT
2	0.000415	0.000395	0.000434	0.00039	0.000416	
4	0.000394	0.000398	0.000396	0.000396	0.000397	
8	0.000448	0.000407	0.000383	0.000389	0.0004	
16	0.000407	0.000395	0.000428	0.00041	0.000425	
32	0.000371	0.0004	0.000414	0.000393	0.000405	
NT						



```
Average GPU time for N=2048, NB=2, NT=2: 0.000404
Average GPU time for N=2048, NB=2, NT=4: 0.000414
Average GPU time for N=2048, NB=2, NT=8: 0.000413
Average GPU time for N=2048, NB=2, NT=16: 0.000396
Average GPU time for N=2048, NB=2, NT=32: 0.000412
Average GPU time for N=2048, NB=4, NT=2: 0.000423
Average GPU time for N=2048, NB=4, NT=4: 0.000408
Average GPU time for N=2048, NB=4, NT=8: 0.000438
Average GPU time for N=2048, NB=4, NT=16: 0.000443
Average GPU time for N=2048, NB=4, NT=32: 0.000440
Average GPU time for N=2048, NB=8, NT=2: 0.000388
Average GPU time for N=2048, NB=8, NT=4: 0.000433
Average GPU time for N=2048, NB=8, NT=8: 0.000417
Average GPU time for N=2048, NB=8, NT=16: 0.000409
Average GPU time for N=2048, NB=8, NT=32: 0.000422
Average GPU time for N=2048, NB=16, NT=2: 0.000408
Average GPU time for N=2048, NB=16, NT=4: 0.000425
Average GPU time for N=2048, NB=16, NT=8: 0.000409
Average GPU time for N=2048, NB=16, NT=16: 0.000428
Average GPU time for N=2048, NB=16, NT=32: 0.000417
Average GPU time for N=2048, NB=32, NT=2: 0.000361
Average GPU time for N=2048, NB=32, NT=4: 0.000399
Average GPU time for N=2048, NB=32, NT=8: 0.000392
Average GPU time for N=2048, NB=32, NT=16: 0.000420
Average GPU time for N=2048, NB=32, NT=32: 0.000411
```

Figure 8 & 9: NB/NT Benchmark (10 Runs per Value)



Sensor	Value	Max
CPU Graphics	0.1 W	0.2 W
CPU DRAM	0.8 W	2.0 W
Generic Memory		
Load		
Memory	66.4 %	68.5 %
Data		
Used Memory	10.5 GB	10.9 GB
Available Memory	5.3 GB	5.4 GB
Total Physical Memory	15.9 GB	15.9 GB
Virtual memory commit l...	21.4 GB	21.4 GB
Virtual memory in use	17.9 GB	18.0 GB
Kernel memory usage	0.5 GB	0.6 GB
Other values		
Processes	351	355
Threads	4691	4850
Handles	144115	145706
NVIDIA NVIDIA GeForce MX150		
Clocks		
GPU Core	139.0 MHz	1645.0 MHz
GPU Memory	405.0 MHz	2505.0 MHz
GPU Shader	278.0 MHz	3290.0 MHz
Temperatures		
✓ GPU Core	60.0 °C	65.0 °C
Load		
✓ GPU Core	0.0 %	82.0 %
GPU Frame Buffer	0.0 %	15.0 %
GPU Video Engine	0.0 %	0.0 %
GPU Bus Interface	0.0 %	45.0 %
GPU Memory	3.2 %	7.0 %
Data		
GPU Memory Free	1982.0 MB	1982.0 MB
GPU Memory Used	65.6 MB	143.7 MB
GPU Memory Total	2048.0 MB	2048.0 MB
Throughput		
GPU PCIE Rx	0.0 KB/s	3471.7 MB/s
GPU PCIE Tx	0.0 KB/s	2006.8 MB/s
SOLIDIGM SSDPFKNU020TZ (C:, ...		
Temperatures		
Temperature	27.0 °C	30.0 °C
Load		
Used Space	52.8 %	52.8 %
Read active time	0.0 %	0.0 %

Figure 10: Throttling behavior and thermal characteristics (1000 Runs per Value, throttling engages within first set of values)

With this in mind it’s difficult to draw conclusively which values of NB and NT are best from the raw data, but it would make sense for higher values to perform better up to a limit. There are hard limitation that are encountered based on GPU capabilities, but within a reasonable range of 2-32, NB = 32 and NT = 32 should have given the best results, if not maybe 16.

Compilation and Testing

As all code was written and tested on my own local device (Nvidia MX150, Ubuntu 22 via WSL), compilation was as simple as *nvcc -o mm_* mm_*.cu*. The submitted code is left in the ideal state for performance benchmarks, but a few lines can be uncommented to display RMSE for all runs and decrease to a limited number of runs.