Machine Problem 2

ELEC 374

Kayne Lee, 20350003

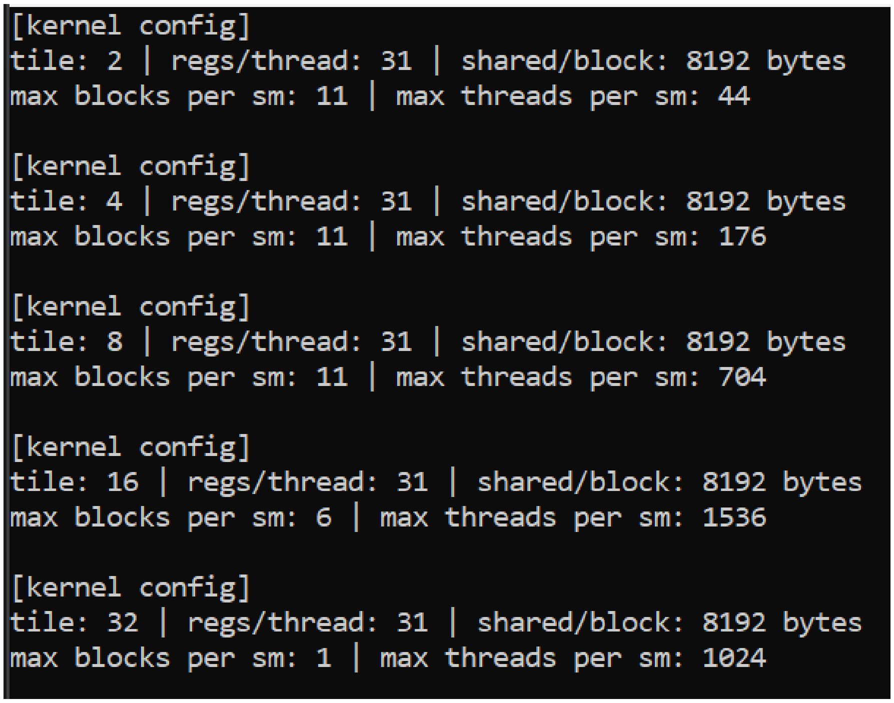
Statement of Originality

“I do hereby verify that this machine problem submission is my own work and contains my own original ideas, concepts, and designs. No portion of this report or code has been copied in whole or in part from another source, with the possible exception of properly referenced material".

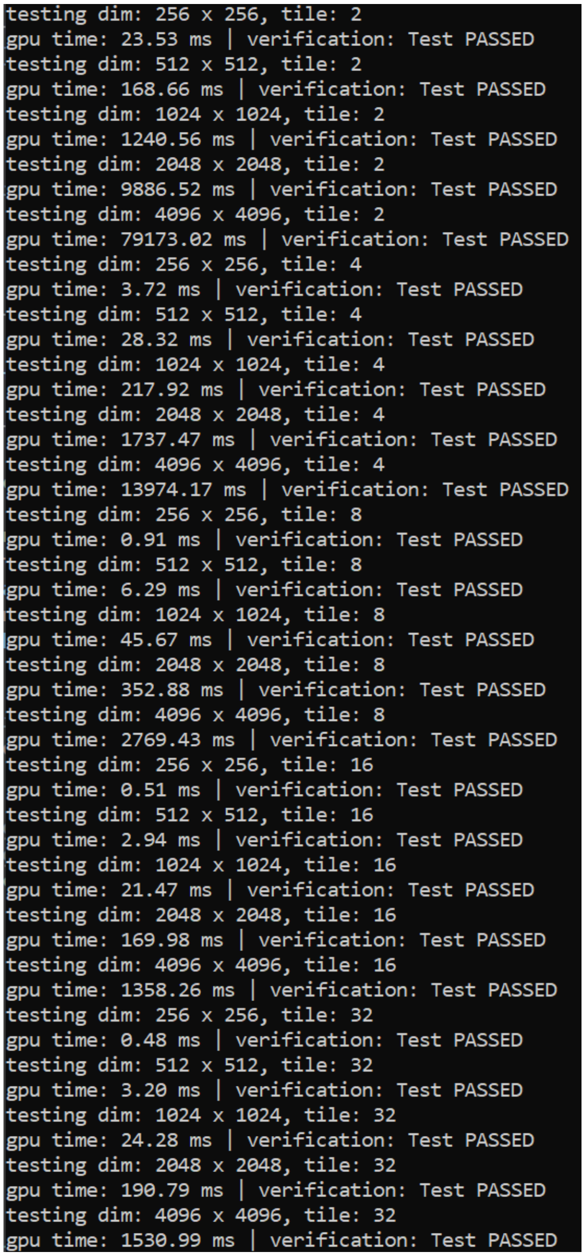
Part 1

The code implements a tiled matrix multiplication algorithm, where both input matrices M and N are multiplied to generate an output matrix P using GPU shared memory. The kernel function handles matrix multiplication in tiles to optimize memory usage and computational efficiency, and the host code initializes random matrices, transfers them to the GPU, invokes the kernel, and verifies the correctness of the GPU result by comparing it to the CPU's result.

Output

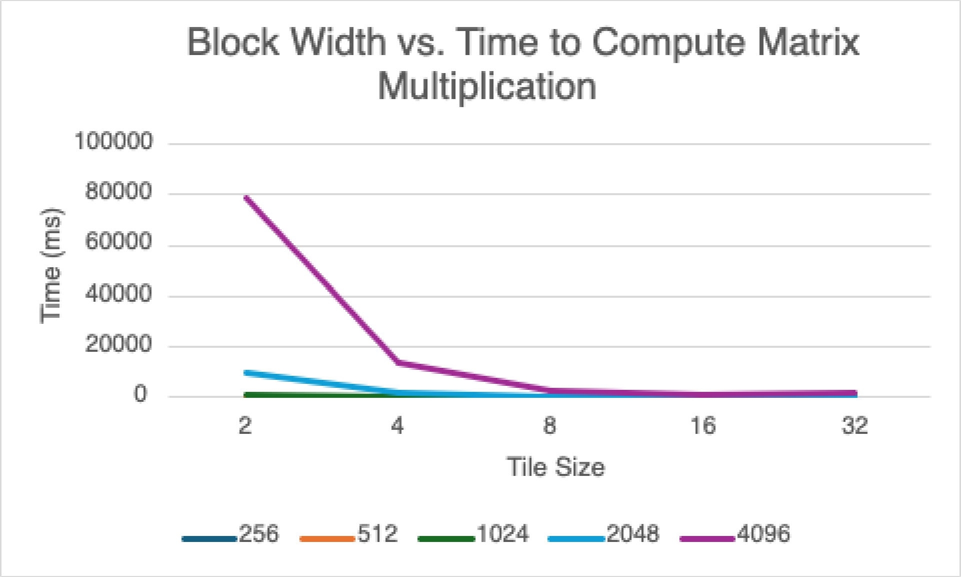


```
[kernel config]
tile: 2 | regs/thread: 31 | shared/block: 8192 bytes
max blocks per sm: 11 | max threads per sm: 44

[kernel config]
tile: 4 | regs/thread: 31 | shared/block: 8192 bytes
max blocks per sm: 11 | max threads per sm: 176

[kernel config]
tile: 8 | regs/thread: 31 | shared/block: 8192 bytes
max blocks per sm: 11 | max threads per sm: 704

[kernel config]
tile: 16 | regs/thread: 31 | shared/block: 8192 bytes
max blocks per sm: 6 | max threads per sm: 1536

[kernel config]
tile: 32 | regs/thread: 31 | shared/block: 8192 bytes
max blocks per sm: 1 | max threads per sm: 1024
```



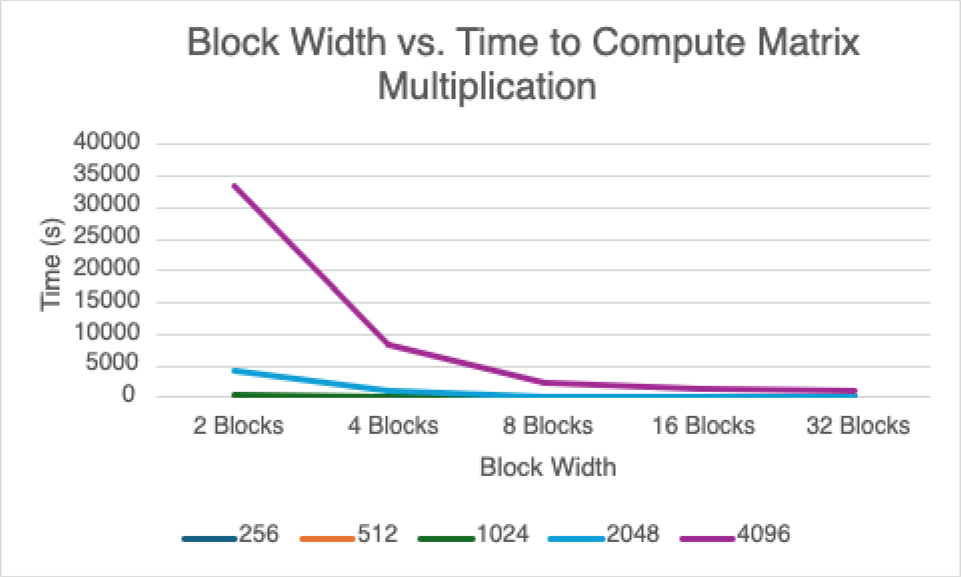
```
testing dim: 256 x 256, tile: 2
gpu time: 23.53 ms | verification: Test PASSED
testing dim: 512 x 512, tile: 2
gpu time: 168.66 ms | verification: Test PASSED
testing dim: 1024 x 1024, tile: 2
gpu time: 1240.56 ms | verification: Test PASSED
testing dim: 2048 x 2048, tile: 2
gpu time: 9886.52 ms | verification: Test PASSED
testing dim: 4096 x 4096, tile: 2
gpu time: 79173.02 ms | verification: Test PASSED
testing dim: 256 x 256, tile: 4
gpu time: 3.72 ms | verification: Test PASSED
testing dim: 512 x 512, tile: 4
gpu time: 28.32 ms | verification: Test PASSED
testing dim: 1024 x 1024, tile: 4
gpu time: 217.92 ms | verification: Test PASSED
testing dim: 2048 x 2048, tile: 4
gpu time: 1737.47 ms | verification: Test PASSED
testing dim: 4096 x 4096, tile: 4
gpu time: 13974.17 ms | verification: Test PASSED
testing dim: 256 x 256, tile: 8
gpu time: 0.91 ms | verification: Test PASSED
testing dim: 512 x 512, tile: 8
gpu time: 6.29 ms | verification: Test PASSED
testing dim: 1024 x 1024, tile: 8
gpu time: 45.67 ms | verification: Test PASSED
testing dim: 2048 x 2048, tile: 8
gpu time: 352.88 ms | verification: Test PASSED
testing dim: 4096 x 4096, tile: 8
gpu time: 2769.43 ms | verification: Test PASSED
testing dim: 256 x 256, tile: 16
gpu time: 0.51 ms | verification: Test PASSED
testing dim: 512 x 512, tile: 16
gpu time: 2.94 ms | verification: Test PASSED
testing dim: 1024 x 1024, tile: 16
gpu time: 21.47 ms | verification: Test PASSED
testing dim: 2048 x 2048, tile: 16
gpu time: 169.98 ms | verification: Test PASSED
testing dim: 4096 x 4096, tile: 16
gpu time: 1358.26 ms | verification: Test PASSED
testing dim: 256 x 256, tile: 32
gpu time: 0.48 ms | verification: Test PASSED
testing dim: 512 x 512, tile: 32
gpu time: 3.20 ms | verification: Test PASSED
testing dim: 1024 x 1024, tile: 32
gpu time: 24.28 ms | verification: Test PASSED
testing dim: 2048 x 2048, tile: 32
gpu time: 190.79 ms | verification: Test PASSED
testing dim: 4096 x 4096, tile: 32
gpu time: 1530.99 ms | verification: Test PASSED
```

Analysis



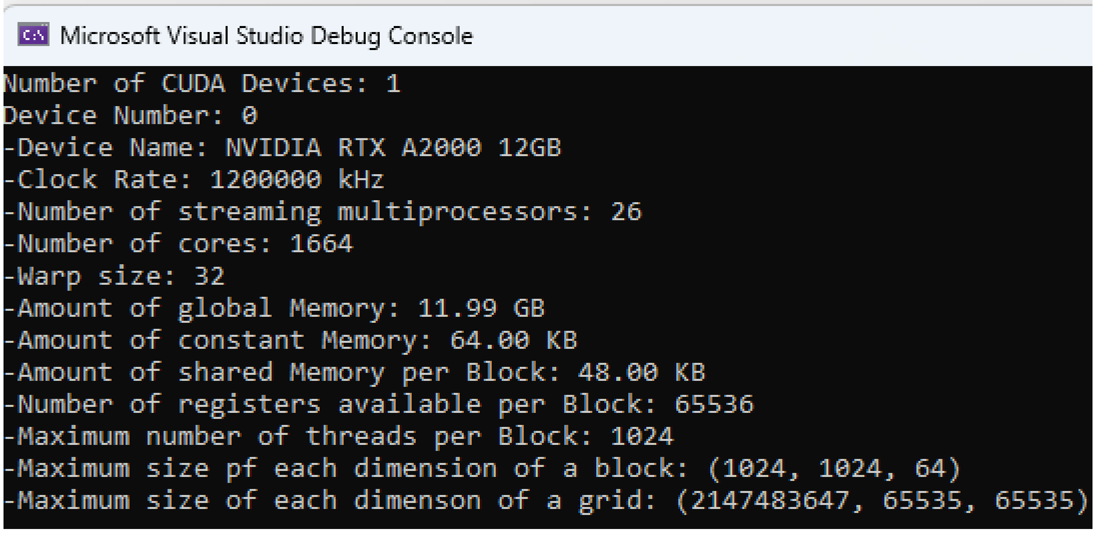
The performance graph demonstrates how computation time for matrix multiplication varies with different tile sizes, highlighting a notable reduction in processing duration as tile dimensions grow from 2×2 to 32×32. This improvement is most pronounced for the largest matrix size tested (4096×4096), which exhibits dramatically faster execution at larger tile sizes compared to smaller matrices. Beyond a tile dimension of 8×8, efficiency gains progressively weaken across all matrix scales, with the performance curve stabilizing to indicate limited returns from further tile size increases.

When comparing to Machine Problem 1 listed here:



Tiled matrix multiplication tends to be slower than standard matrix multiplication, likely due to the additional overhead introduced by tiling. In contrast, normal matrix multiplication is faster because it follows a simpler approach without this extra computation.

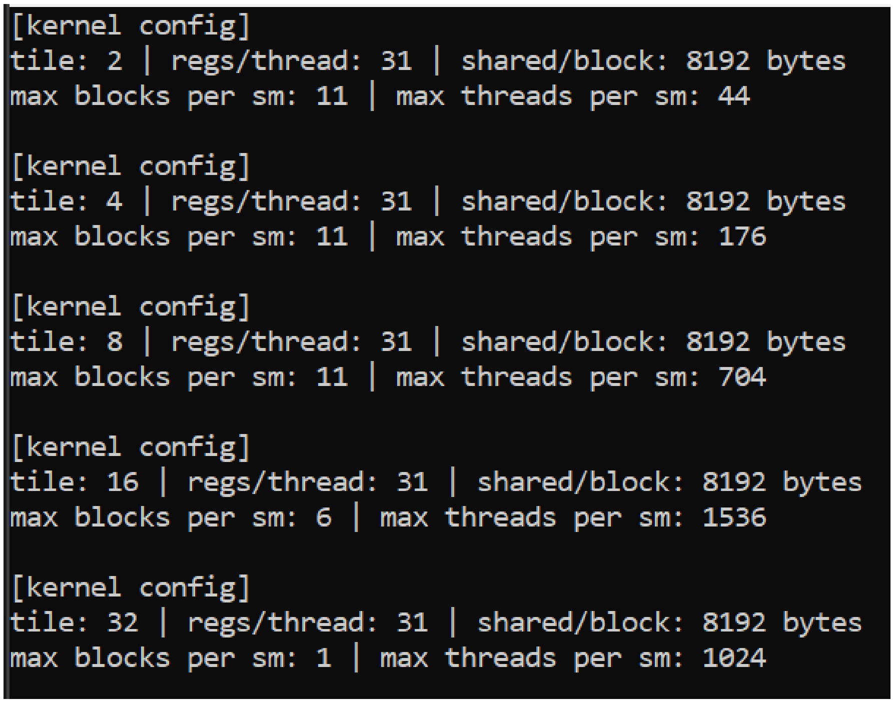
a)



```
Number of CUDA Devices: 1
Device Number: 0
-Device Name: NVIDIA RTX A2000 12GB
-Clock Rate: 1200000 kHz
-Number of streaming multiprocessors: 26
-Number of cores: 1664
-Warp size: 32
-Amount of global Memory: 11.99 GB
-Amount of constant Memory: 64.00 KB
-Amount of shared Memory per Block: 48.00 KB
-Number of registers available per Block: 65536
-Maximum number of threads per Block: 1024
-Maximum size pf each dimension of a block: (1024, 1024, 64)
-Maximum size of each dimenson of a grid: (2147483647, 65535, 65535)
```

The total number of threads that can be scheduled concurrently on a CUDA device is based on the number of streaming multiprocessors (SMs) and the maximum threads each SM can handle, calculated using this formula: Total Threads = SMs x Maximum Threads per SM.

This gives us a total of 26 SMs multiplied by 1024 maximum threads per SM, resulting in 26,624 threads.



```
[kernel config]
tile: 2 | regs/thread: 31 | shared/block: 8192 bytes
max blocks per sm: 11 | max threads per sm: 44

[kernel config]
tile: 4 | regs/thread: 31 | shared/block: 8192 bytes
max blocks per sm: 11 | max threads per sm: 176

[kernel config]
tile: 8 | regs/thread: 31 | shared/block: 8192 bytes
max blocks per sm: 11 | max threads per sm: 704

[kernel config]
tile: 16 | regs/thread: 31 | shared/block: 8192 bytes
max blocks per sm: 6 | max threads per sm: 1536

[kernel config]
tile: 32 | regs/thread: 31 | shared/block: 8192 bytes
max blocks per sm: 1 | max threads per sm: 1024
```

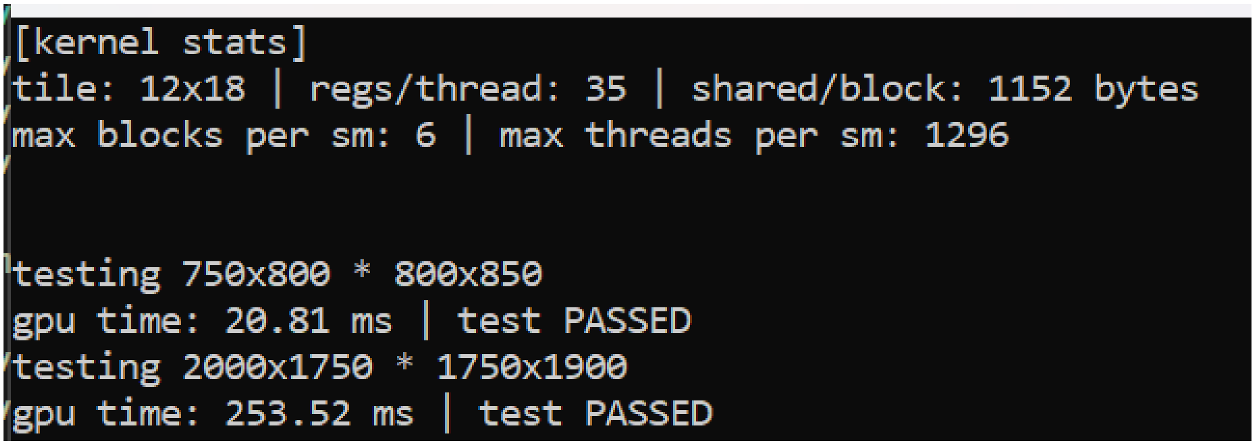
b) Resource usage of the kernel was found using the cudaFuncGetAttributes and cudaOccupancyMaxActiveBlocksPerMultiprocessor.

- 2x2: each thread uses 31 registers, shared memory size is 8192 bytes, number of blocks per streaming multiprocessor is 11, and max number of threads is 44
- 4x4: each thread uses 31 registers, shared memory size is 8192 bytes, number of blocks per streaming multiprocessor is 11, and max number of threads is 176
- 8x8: each thread uses 31 registers, shared memory size is 8192 bytes, number of blocks per streaming multiprocessor is 11, and max number of threads is 704
- 16x16: each thread uses 31 registers, shared memory size is 8192 bytes, number of blocks per streaming multiprocessor is 6, and max number of threads is 1536
- 32x32: each thread uses 31 registers, shared memory size is 8192 bytes, number of blocks per streaming multiprocessor is 1, and max number of threads is 1024

Part 2

This part consists of analyzing memory transfer time compared to matrix size, GPU vs CPU time to compute matrix multiplication, and GPU block size vs time to compute matrix multiplication.

Output



```
[kernel stats]
tile: 12x18 | regs/thread: 35 | shared/block: 1152 bytes
max blocks per sm: 6 | max threads per sm: 1296


testing 750x800 * 800x850
gpu time: 20.81 ms | test PASSED
testing 2000x1750 * 1750x1900
gpu time: 253.52 ms | test PASSED
```

Analysis

This is the output from the GPU revised tile matrix multiplication compared to CPU matrix multiplication implementation. This shows that tiled matrix multiplication is much more efficient than CPU.