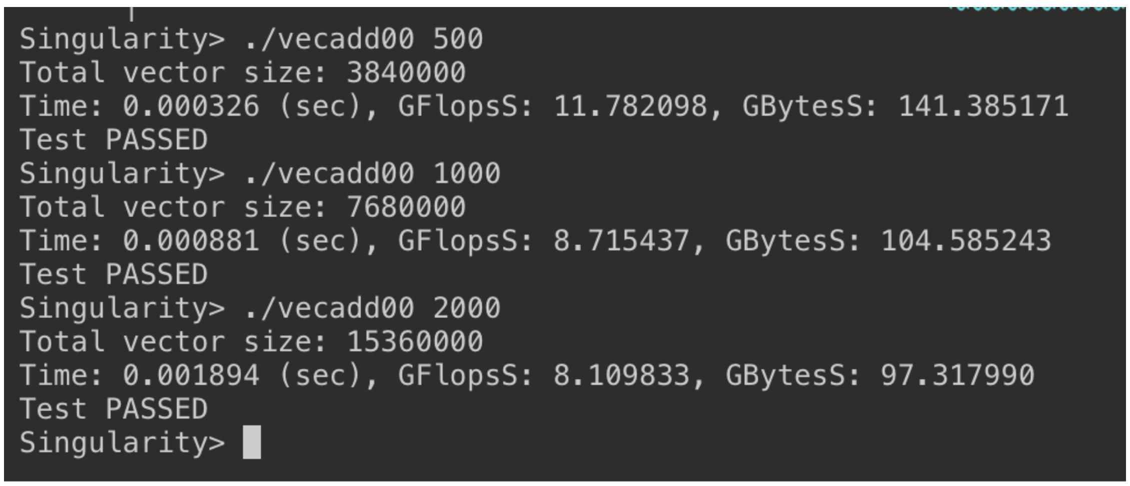
HPML Homework 4

Name: Jayraj Pamnani
Netid: jmp10051

Part A

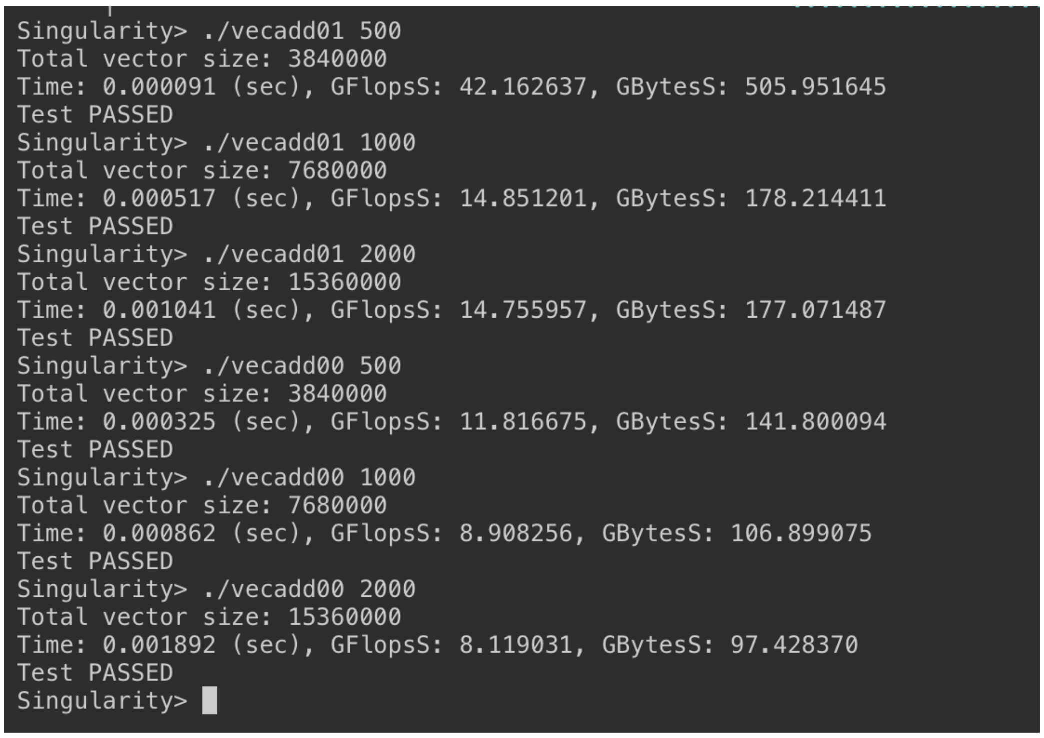
Problem-1

Q1



```
Singularity> ./vecadd00 500
Total vector size: 3840000
Time: 0.000326 (sec), GFlopsS: 11.782098, GBytesS: 141.385171
Test PASSED
Singularity> ./vecadd00 1000
Total vector size: 7680000
Time: 0.000881 (sec), GFlopsS: 8.715437, GBytesS: 104.585243
Test PASSED
Singularity> ./vecadd00 2000
Total vector size: 15360000
Time: 0.001894 (sec), GFlopsS: 8.109833, GBytesS: 97.317990
Test PASSED
Singularity>
```

Q2



```
Singularity> ./vecadd01 500
Total vector size: 3840000
Time: 0.000091 (sec), GFlopsS: 42.162637, GBytesS: 505.951645
Test PASSED
Singularity> ./vecadd01 1000
Total vector size: 7680000
Time: 0.000517 (sec), GFlopsS: 14.851201, GBytesS: 178.214411
Test PASSED
Singularity> ./vecadd01 2000
Total vector size: 15360000
Time: 0.001041 (sec), GFlopsS: 14.755957, GBytesS: 177.071487
Test PASSED
Singularity> ./vecadd00 500
Total vector size: 3840000
Time: 0.000325 (sec), GFlopsS: 11.816675, GBytesS: 141.800094
Test PASSED
Singularity> ./vecadd00 1000
Total vector size: 7680000
Time: 0.000862 (sec), GFlopsS: 8.908256, GBytesS: 106.899075
Test PASSED
Singularity> ./vecadd00 2000
Total vector size: 15360000
Time: 0.001892 (sec), GFlopsS: 8.119031, GBytesS: 97.428370
Test PASSED
Singularity>
```

Observations

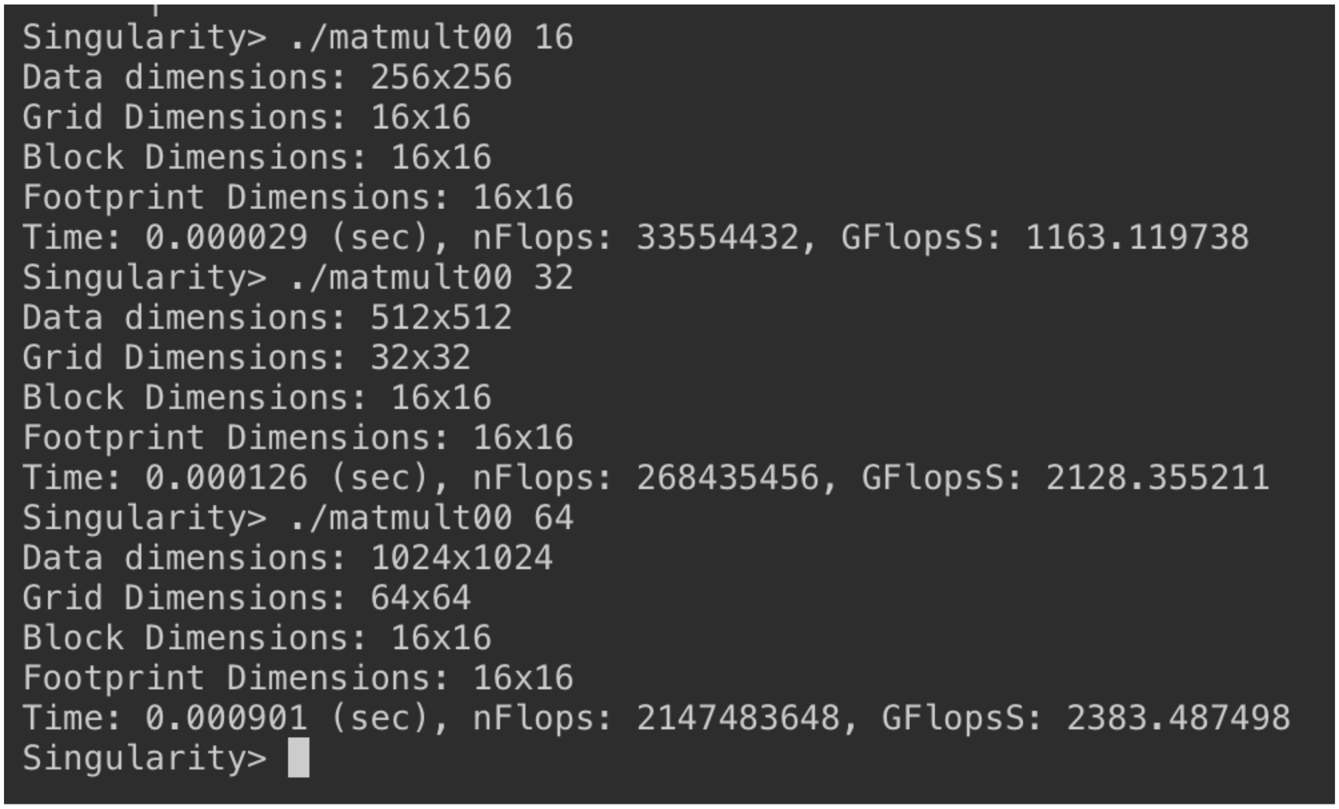
1. Coalesced access is faster across all sizes, with the largest improvement at 500 (about 3.6x).
2. The coalesced version shows better memory bandwidth utilization (higher GBytesS).
3. Both versions show decreasing GFlops as size increases, likely due to memory bandwidth limits.
4. All tests passed, confirming correctness.

Why coalesced access is faster

- Non-coalesced (vecadd00): threads access memory with stride N, causing multiple separate memory transactions.
- Coalesced (vecadd01): consecutive threads access consecutive memory locations, allowing the GPU to combine accesses into fewer, wider transactions.

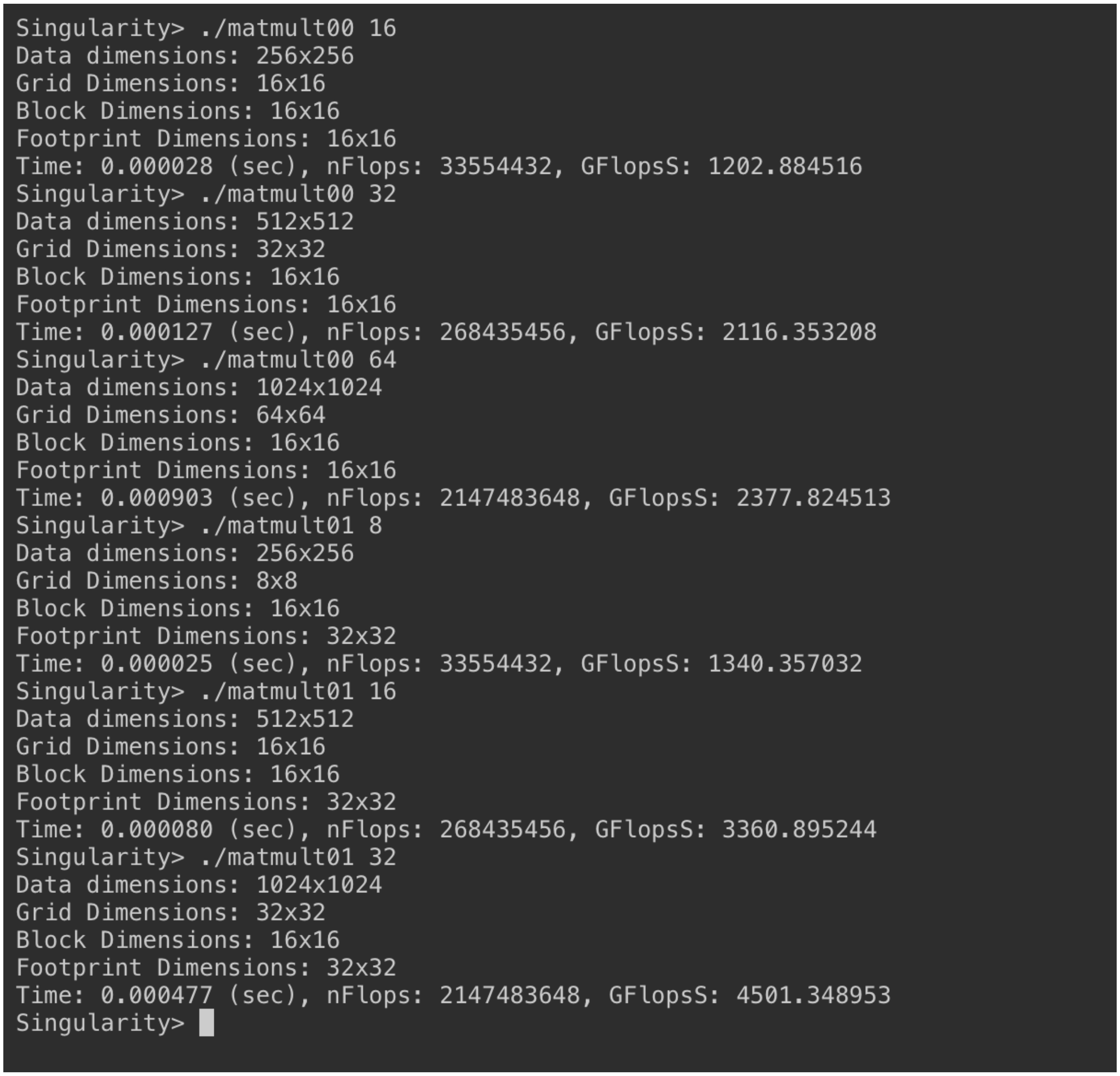
Problem 2

Q3



```
Singularity> ./matmult00 16
Data dimensions: 256x256
Grid Dimensions: 16x16
Block Dimensions: 16x16
Footprint Dimensions: 16x16
Time: 0.000029 (sec), nFlops: 33554432, GFlopsS: 1163.119738
Singularity> ./matmult00 32
Data dimensions: 512x512
Grid Dimensions: 32x32
Block Dimensions: 16x16
Footprint Dimensions: 16x16
Time: 0.000126 (sec), nFlops: 268435456, GFlopsS: 2128.355211
Singularity> ./matmult00 64
Data dimensions: 1024x1024
Grid Dimensions: 64x64
Block Dimensions: 16x16
Footprint Dimensions: 16x16
Time: 0.000901 (sec), nFlops: 2147483648, GFlopsS: 2383.487498
Singularity> 
```

Q4



```
Singularity> ./matmult00 16
Data dimensions: 256x256
Grid Dimensions: 16x16
Block Dimensions: 16x16
Footprint Dimensions: 16x16
Time: 0.000028 (sec), nFlops: 33554432, GFlopsS: 1202.884516
Singularity> ./matmult00 32
Data dimensions: 512x512
Grid Dimensions: 32x32
Block Dimensions: 16x16
Footprint Dimensions: 16x16
Time: 0.000127 (sec), nFlops: 268435456, GFlopsS: 2116.353208
Singularity> ./matmult00 64
Data dimensions: 1024x1024
Grid Dimensions: 64x64
Block Dimensions: 16x16
Footprint Dimensions: 16x16
Time: 0.000903 (sec), nFlops: 2147483648, GFlopsS: 2377.824513
Singularity> ./matmult01 8
Data dimensions: 256x256
Grid Dimensions: 8x8
Block Dimensions: 16x16
Footprint Dimensions: 32x32
Time: 0.000025 (sec), nFlops: 33554432, GFlopsS: 1340.357032
Singularity> ./matmult01 16
Data dimensions: 512x512
Grid Dimensions: 16x16
Block Dimensions: 16x16
Footprint Dimensions: 32x32
Time: 0.000080 (sec), nFlops: 268435456, GFlopsS: 3360.895244
Singularity> ./matmult01 32
Data dimensions: 1024x1024
Grid Dimensions: 32x32
Block Dimensions: 16x16
Footprint Dimensions: 32x32
Time: 0.000477 (sec), nFlops: 2147483648, GFlopsS: 4501.348953
Singularity> 
```

Performance Analysis

Performance comparison

Matrix Size	matmult00 (16×16)	matmult01 (32×32)	Improvement
256×256	1202.88 GFlops/s	1340.36 GFlops/s	+11.4%

Matrix Size	**matmult00 (16×16)**	**matmult01 (32×32)**	**Improvement**
512×512	2116.35 GFlops/s	3360.90 GFlops/s	+58.8%
1024×1024	2377.82 GFlops/s	4501.35 GFlops/s	+89.3%

Observations

1. Larger improvements at larger sizes:
   - 256×256: ~11% faster
   - 512×512: ~59% faster
   - 1024×1024: ~89% faster
2. Grid size reduction:
   - 256×256: 16×16 → 8×8 (4× fewer blocks)
   - 512×512: 32×32 → 16×16 (4× fewer blocks)
   - 1024×1024: 64×64 → 32×32 (4× fewer blocks)
3. Better scaling with size: matmult01 continues to improve at 1024×1024, while matmult00 appears to plateau.

Why the optimization works

1. Reduced grid overhead:
   - Fewer thread blocks → less launch and synchronization overhead
   - 4× fewer blocks with the 32×32 footprint
2. Better resource utilization:
   - Each thread computes 4 elements (2×2) instead of 1
   - Better register reuse and instruction-level parallelism
3. Improved memory access patterns:
   - Larger shared memory tiles (32×33) reduce global memory traffic
   - Better cache locality with fewer blocks
4. Loop unrolling:
   - #pragma unroll helps the compiler optimize the inner product loops

Q5

Rules of Thumb:

1. Memory Access Coalescing — Consecutive threads should access consecutive memory locations

2. Optimize Block and Footprint Sizes — Larger footprints (e.g., 32×32) reduce grid overhead for larger problems
3. Increase Work per Thread — Compute multiple elements per thread to better utilize registers
4. Use Shared Memory Strategically — Cache and reuse data, pad to avoid bank conflicts
5. Minimize Grid Overhead — Fewer thread blocks reduce launch/sync overhead
6. Performance Scales with Problem Size — Optimizations show larger gains on larger problems
7. Use Loop Unrolling — Helps compiler optimize inner loops
8. Balance Occupancy and Resource Usage — Higher occupancy is good, but not always necessary

The document is saved as CONCLUSIONS.md in your PartA directory. You can:

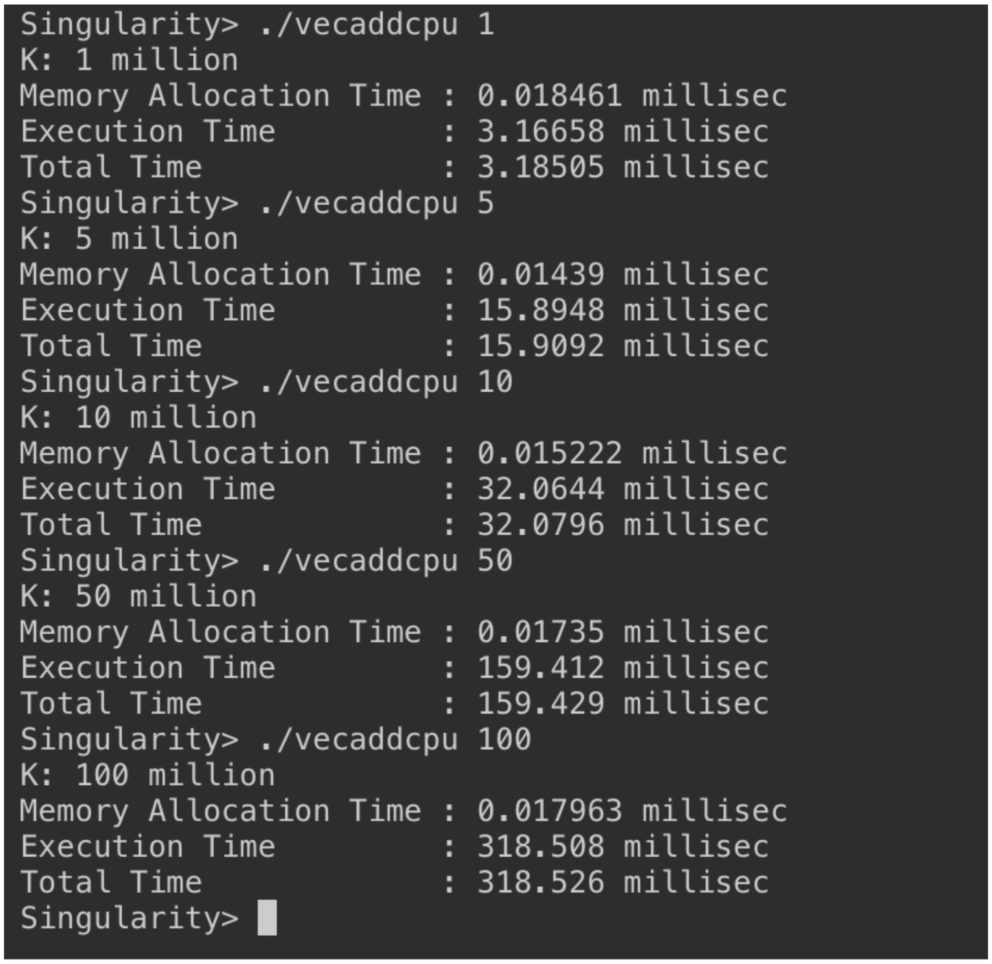
- Copy it into your lab report
- Customize it based on your specific results
- Add specific data points from your experiments

The conclusions are based on:

- Your vector addition results (coalescing vs non-coalescing)
- Your matrix multiplication results (32×32 vs 16×16 footprint)
- The performance improvements you observed (11% to 89% improvement)

Part B

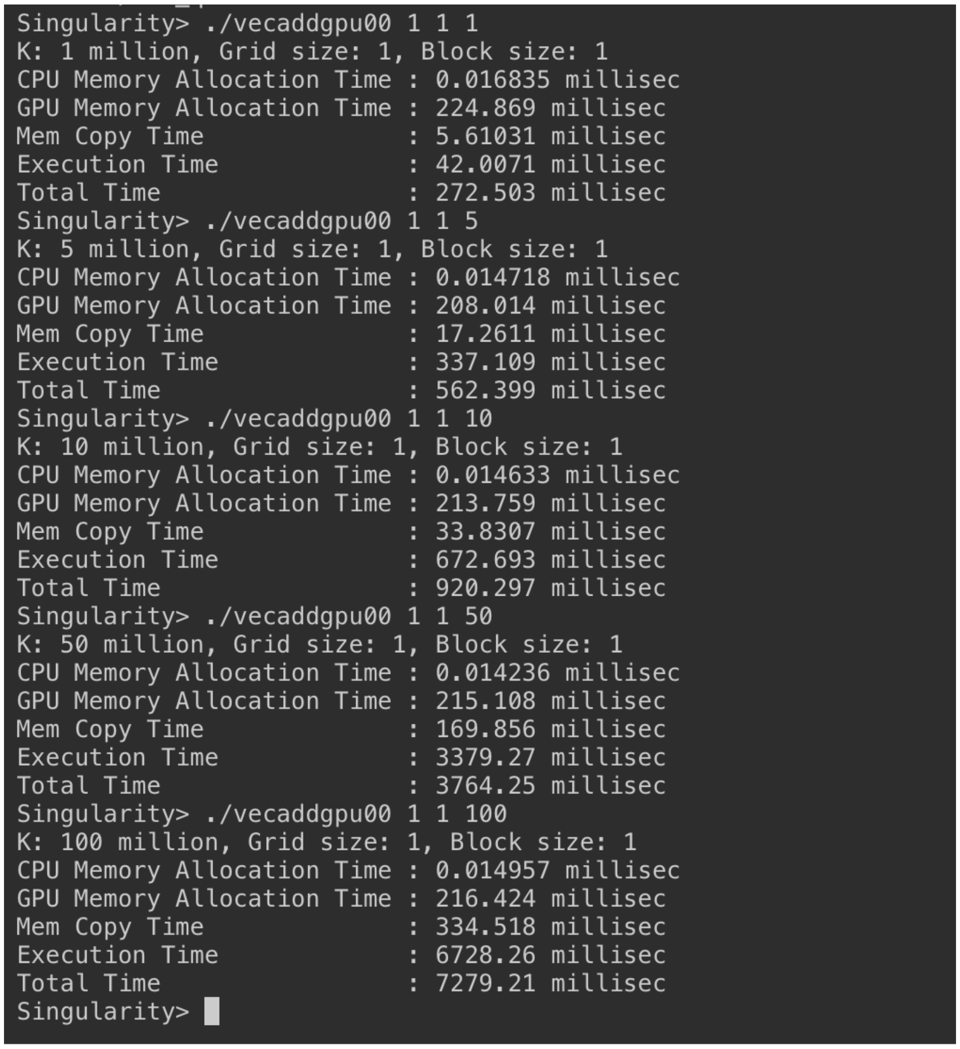
Q1



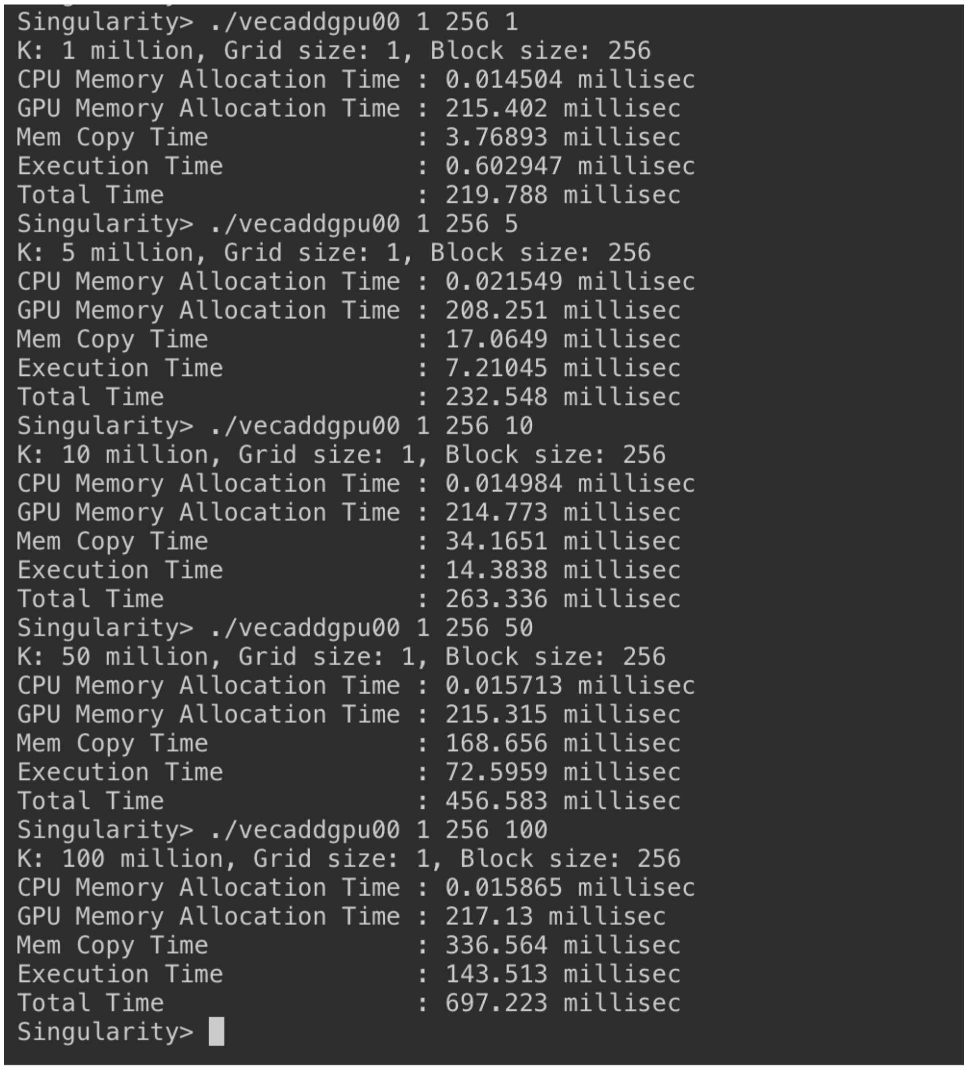
```
Singularity> ./vecaddcpu 1
K: 1 million
Memory Allocation Time : 0.018461 millisec
Execution Time         : 3.16658 millisec
Total Time             : 3.18505 millisec
Singularity> ./vecaddcpu 5
K: 5 million
Memory Allocation Time : 0.01439 millisec
Execution Time         : 15.8948 millisec
Total Time             : 15.9092 millisec
Singularity> ./vecaddcpu 10
K: 10 million
Memory Allocation Time : 0.015222 millisec
Execution Time         : 32.0644 millisec
Total Time             : 32.0796 millisec
Singularity> ./vecaddcpu 50
K: 50 million
Memory Allocation Time : 0.01735 millisec
Execution Time         : 159.412 millisec
Total Time             : 159.429 millisec
Singularity> ./vecaddcpu 100
K: 100 million
Memory Allocation Time : 0.017963 millisec
Execution Time         : 318.508 millisec
Total Time             : 318.526 millisec
Singularity>
```

Q2

Scenario 1: Using one block with 1 thread

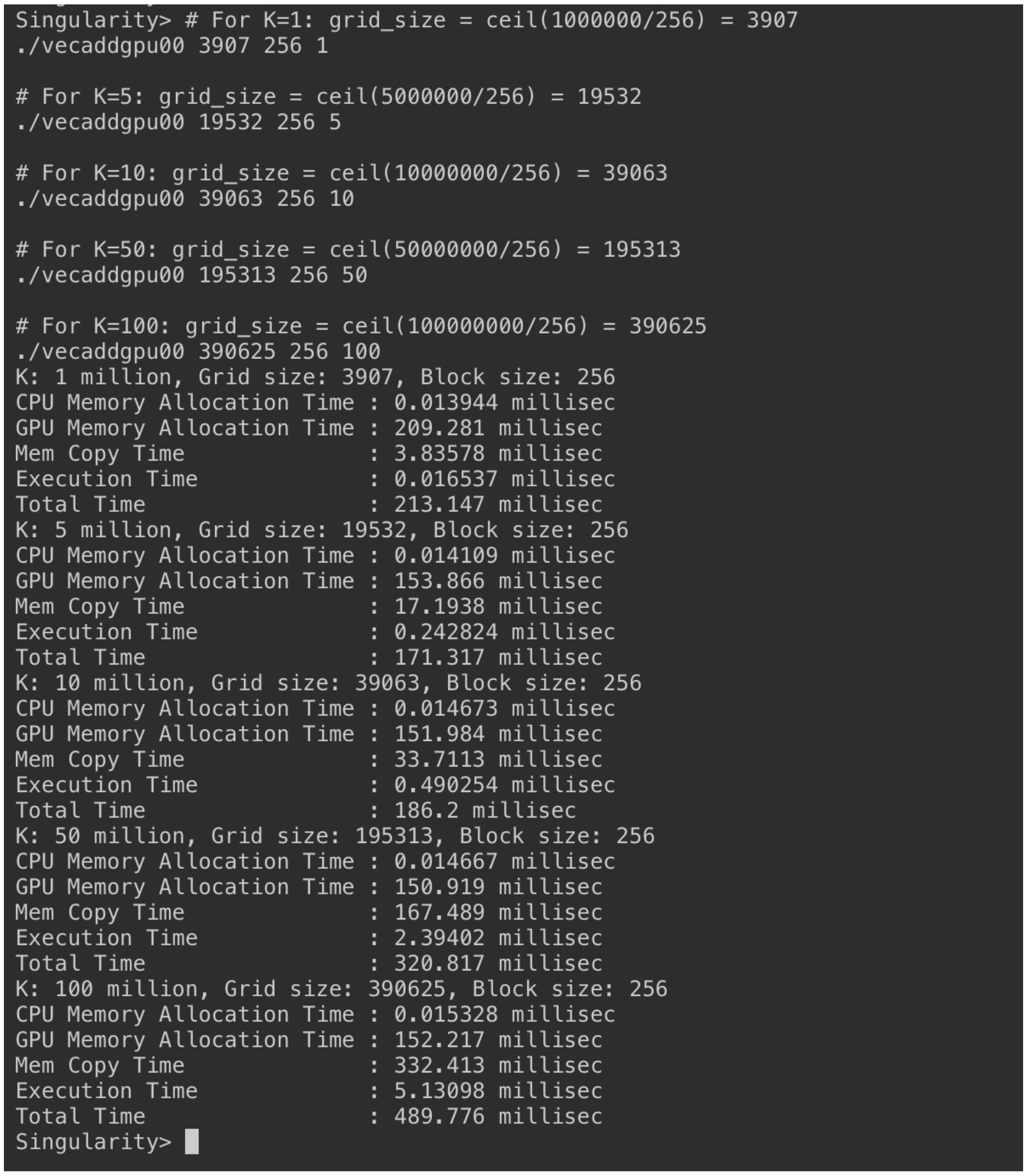

```
Singularity> ./vecaddgpu00 1 1 1
K: 1 million, Grid size: 1, Block size: 1
CPU Memory Allocation Time : 0.016835 millisec
GPU Memory Allocation Time : 224.869 millisec
Mem Copy Time              : 5.61031 millisec
Execution Time             : 42.0071 millisec
Total Time                 : 272.503 millisec
Singularity> ./vecaddgpu00 1 1 5
K: 5 million, Grid size: 1, Block size: 1
CPU Memory Allocation Time : 0.014718 millisec
GPU Memory Allocation Time : 208.014 millisec
Mem Copy Time              : 17.2611 millisec
Execution Time             : 337.109 millisec
Total Time                 : 562.399 millisec
Singularity> ./vecaddgpu00 1 1 10
K: 10 million, Grid size: 1, Block size: 1
CPU Memory Allocation Time : 0.014633 millisec
GPU Memory Allocation Time : 213.759 millisec
Mem Copy Time              : 33.8307 millisec
Execution Time             : 672.693 millisec
Total Time                 : 920.297 millisec
Singularity> ./vecaddgpu00 1 1 50
K: 50 million, Grid size: 1, Block size: 1
CPU Memory Allocation Time : 0.014236 millisec
GPU Memory Allocation Time : 215.108 millisec
Mem Copy Time              : 169.856 millisec
Execution Time             : 3379.27 millisec
Total Time                 : 3764.25 millisec
Singularity> ./vecaddgpu00 1 1 100
K: 100 million, Grid size: 1, Block size: 1
CPU Memory Allocation Time : 0.014957 millisec
GPU Memory Allocation Time : 216.424 millisec
Mem Copy Time              : 334.518 millisec
Execution Time             : 6728.26 millisec
Total Time                 : 7279.21 millisec
Singularity>
```

Scenario 2: Using one block with 256 threads



```
Singularity> ./vecaddgpu00 1 256 1
K: 1 million, Grid size: 1, Block size: 256
CPU Memory Allocation Time : 0.014504 millisec
GPU Memory Allocation Time : 215.402 millisec
Mem Copy Time              : 3.76893 millisec
Execution Time             : 0.602947 millisec
Total Time                 : 219.788 millisec
Singularity> ./vecaddgpu00 1 256 5
K: 5 million, Grid size: 1, Block size: 256
CPU Memory Allocation Time : 0.021549 millisec
GPU Memory Allocation Time : 208.251 millisec
Mem Copy Time              : 17.0649 millisec
Execution Time             : 7.21045 millisec
Total Time                 : 232.548 millisec
Singularity> ./vecaddgpu00 1 256 10
K: 10 million, Grid size: 1, Block size: 256
CPU Memory Allocation Time : 0.014984 millisec
GPU Memory Allocation Time : 214.773 millisec
Mem Copy Time              : 34.1651 millisec
Execution Time             : 14.3838 millisec
Total Time                 : 263.336 millisec
Singularity> ./vecaddgpu00 1 256 50
K: 50 million, Grid size: 1, Block size: 256
CPU Memory Allocation Time : 0.015713 millisec
GPU Memory Allocation Time : 215.315 millisec
Mem Copy Time              : 168.656 millisec
Execution Time             : 72.5959 millisec
Total Time                 : 456.583 millisec
Singularity> ./vecaddgpu00 1 256 100
K: 100 million, Grid size: 1, Block size: 256
CPU Memory Allocation Time : 0.015865 millisec
GPU Memory Allocation Time : 217.13 millisec
Mem Copy Time              : 336.564 millisec
Execution Time             : 143.513 millisec
Total Time                 : 697.223 millisec
Singularity>
```

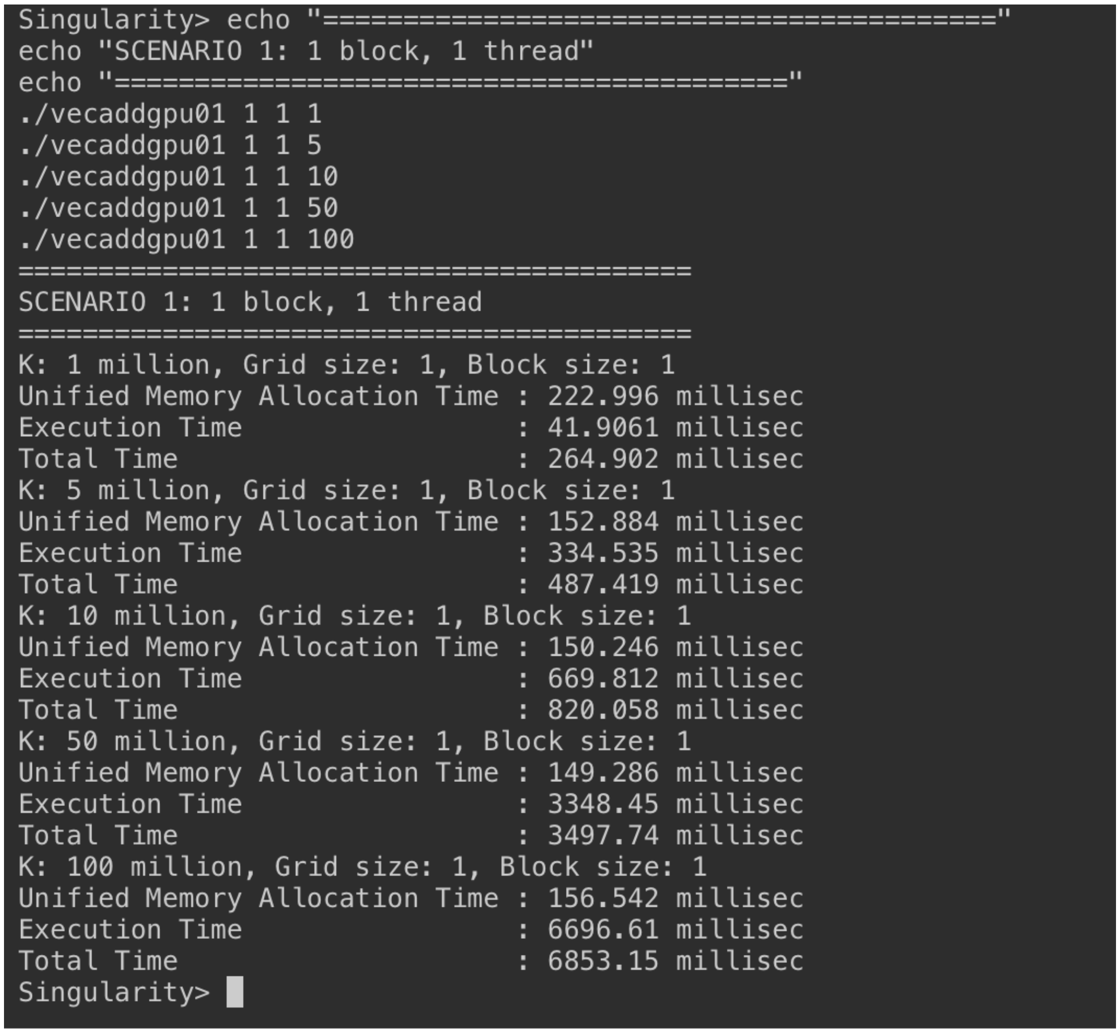
Scenario 3: Using multiple blocks with 256 threads per block, with the total number of threads
across all blocks equal to the size of arrays.



```
Singularity> # For K=1: grid_size = ceil(1000000/256) = 3907
./vecaddgpu00 3907 256 1

# For K=5: grid_size = ceil(5000000/256) = 19532
./vecaddgpu00 19532 256 5

# For K=10: grid_size = ceil(10000000/256) = 39063
./vecaddgpu00 39063 256 10

# For K=50: grid_size = ceil(50000000/256) = 195313
./vecaddgpu00 195313 256 50

# For K=100: grid_size = ceil(100000000/256) = 390625
./vecaddgpu00 390625 256 100
K: 1 million, Grid size: 3907, Block size: 256
CPU Memory Allocation Time : 0.013944 millisec
GPU Memory Allocation Time : 209.281 millisec
Mem Copy Time              : 3.83578 millisec
Execution Time             : 0.016537 millisec
Total Time                 : 213.147 millisec
K: 5 million, Grid size: 19532, Block size: 256
CPU Memory Allocation Time : 0.014109 millisec
GPU Memory Allocation Time : 153.866 millisec
Mem Copy Time              : 17.1938 millisec
Execution Time             : 0.242824 millisec
Total Time                 : 171.317 millisec
K: 10 million, Grid size: 39063, Block size: 256
CPU Memory Allocation Time : 0.014673 millisec
GPU Memory Allocation Time : 151.984 millisec
Mem Copy Time              : 33.7113 millisec
Execution Time             : 0.490254 millisec
Total Time                 : 186.2 millisec
K: 50 million, Grid size: 195313, Block size: 256
CPU Memory Allocation Time : 0.014667 millisec
GPU Memory Allocation Time : 150.919 millisec
Mem Copy Time              : 167.489 millisec
Execution Time             : 2.39402 millisec
Total Time                 : 320.817 millisec
K: 100 million, Grid size: 390625, Block size: 256
CPU Memory Allocation Time : 0.015328 millisec
GPU Memory Allocation Time : 152.217 millisec
Mem Copy Time              : 332.413 millisec
Execution Time             : 5.13098 millisec
Total Time                 : 489.776 millisec
Singularity>
```

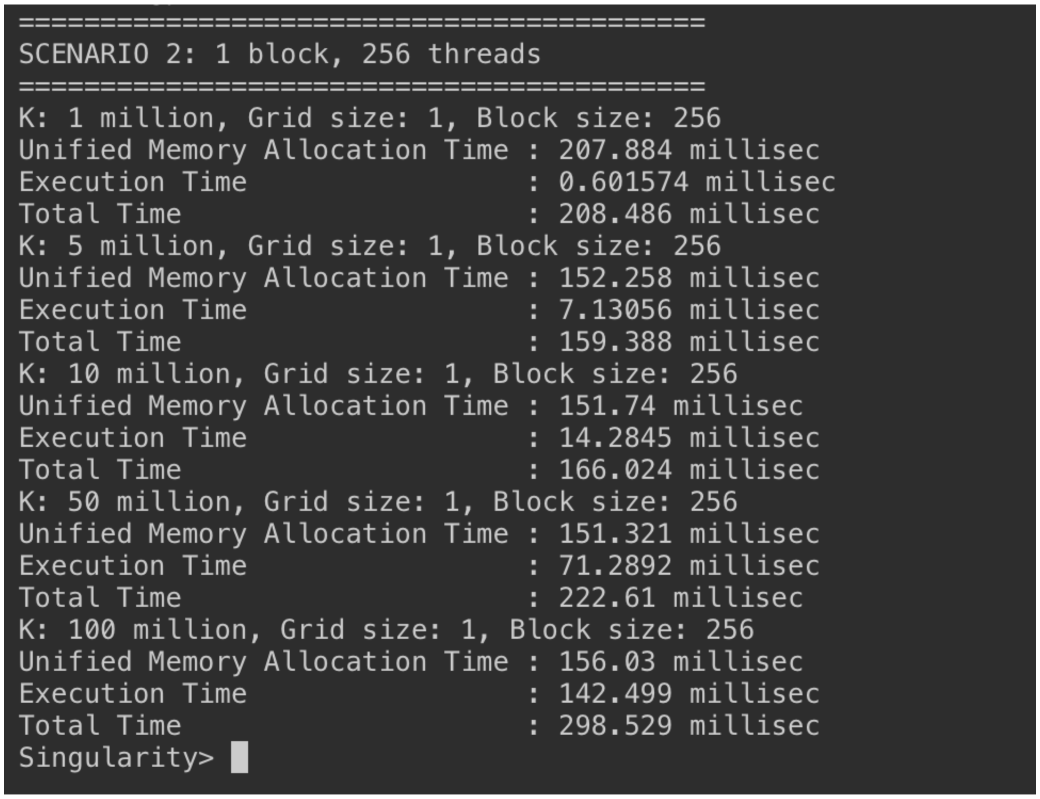
Q3

Scenario 1



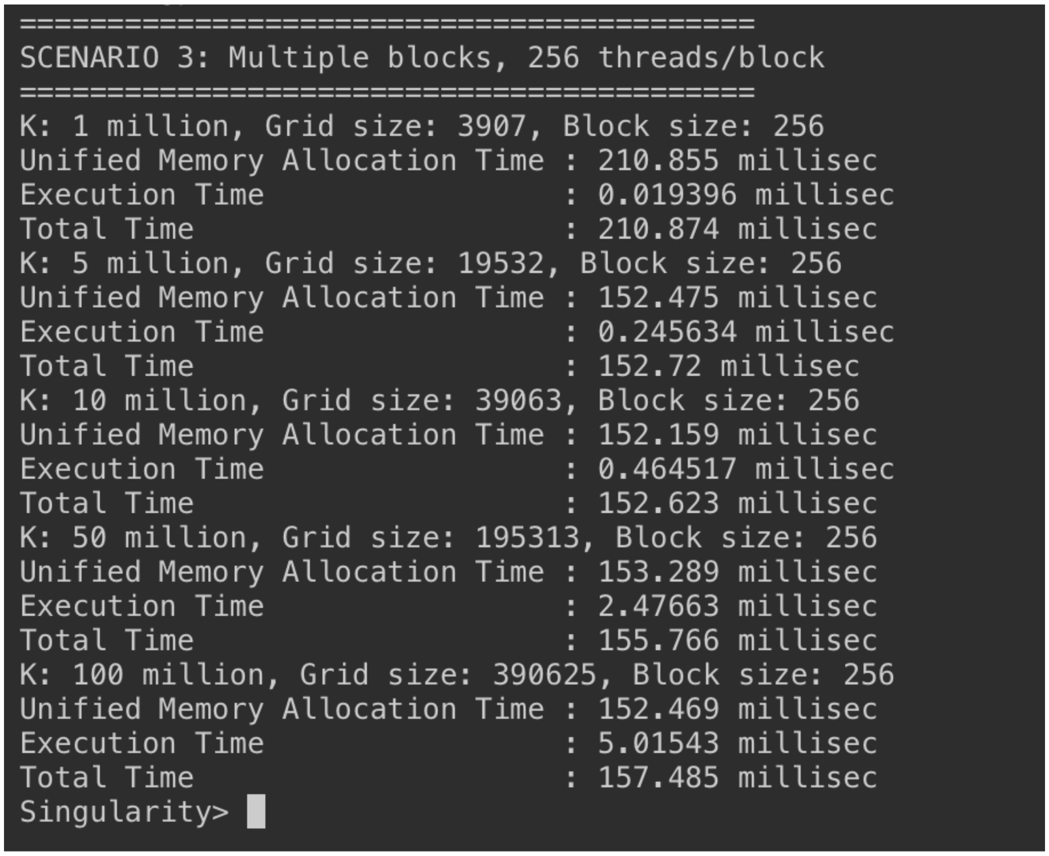
```
Singularity> echo "==========================================="
echo "SCENARIO 1: 1 block, 1 thread"
echo "==========================================="
./vecaddgpu01 1 1 1
./vecaddgpu01 1 1 5
./vecaddgpu01 1 1 10
./vecaddgpu01 1 1 50
./vecaddgpu01 1 1 100
==========================================
SCENARIO 1: 1 block, 1 thread
==========================================
K: 1 million, Grid size: 1, Block size: 1
Unified Memory Allocation Time : 222.996 millisec
Execution Time                 : 41.9061 millisec
Total Time                     : 264.902 millisec
K: 5 million, Grid size: 1, Block size: 1
Unified Memory Allocation Time : 152.884 millisec
Execution Time                 : 334.535 millisec
Total Time                     : 487.419 millisec
K: 10 million, Grid size: 1, Block size: 1
Unified Memory Allocation Time : 150.246 millisec
Execution Time                 : 669.812 millisec
Total Time                     : 820.058 millisec
K: 50 million, Grid size: 1, Block size: 1
Unified Memory Allocation Time : 149.286 millisec
Execution Time                 : 3348.45 millisec
Total Time                     : 3497.74 millisec
K: 100 million, Grid size: 1, Block size: 1
Unified Memory Allocation Time : 156.542 millisec
Execution Time                 : 6696.61 millisec
Total Time                     : 6853.15 millisec
Singularity>
```

Scenario 2



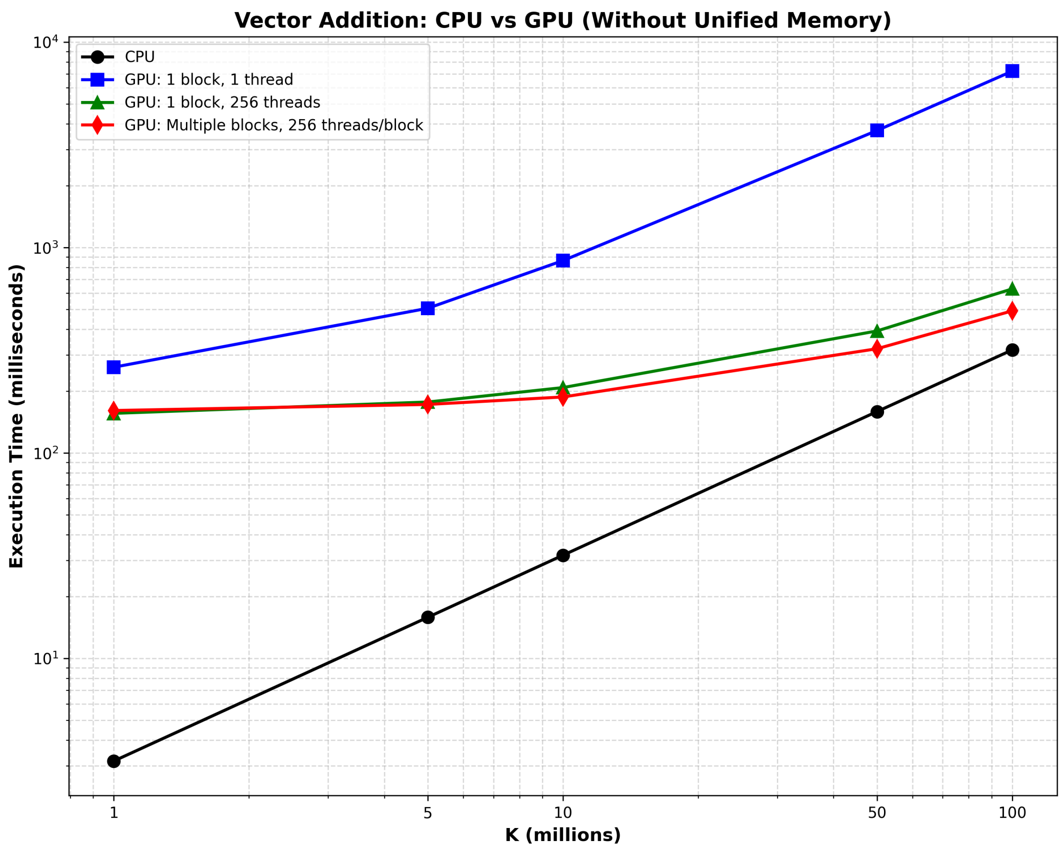
```
==========================================
SCENARIO 2: 1 block, 256 threads
==========================================
K: 1 million, Grid size: 1, Block size: 256
Unified Memory Allocation Time : 207.884 millisec
Execution Time                 : 0.601574 millisec
Total Time                     : 208.486 millisec
K: 5 million, Grid size: 1, Block size: 256
Unified Memory Allocation Time : 152.258 millisec
Execution Time                 : 7.13056 millisec
Total Time                     : 159.388 millisec
K: 10 million, Grid size: 1, Block size: 256
Unified Memory Allocation Time : 151.74 millisec
Execution Time                 : 14.2845 millisec
Total Time                     : 166.024 millisec
K: 50 million, Grid size: 1, Block size: 256
Unified Memory Allocation Time : 151.321 millisec
Execution Time                 : 71.2892 millisec
Total Time                     : 222.61 millisec
K: 100 million, Grid size: 1, Block size: 256
Unified Memory Allocation Time : 156.03 millisec
Execution Time                 : 142.499 millisec
Total Time                     : 298.529 millisec
Singularity>
```

Scenario 3

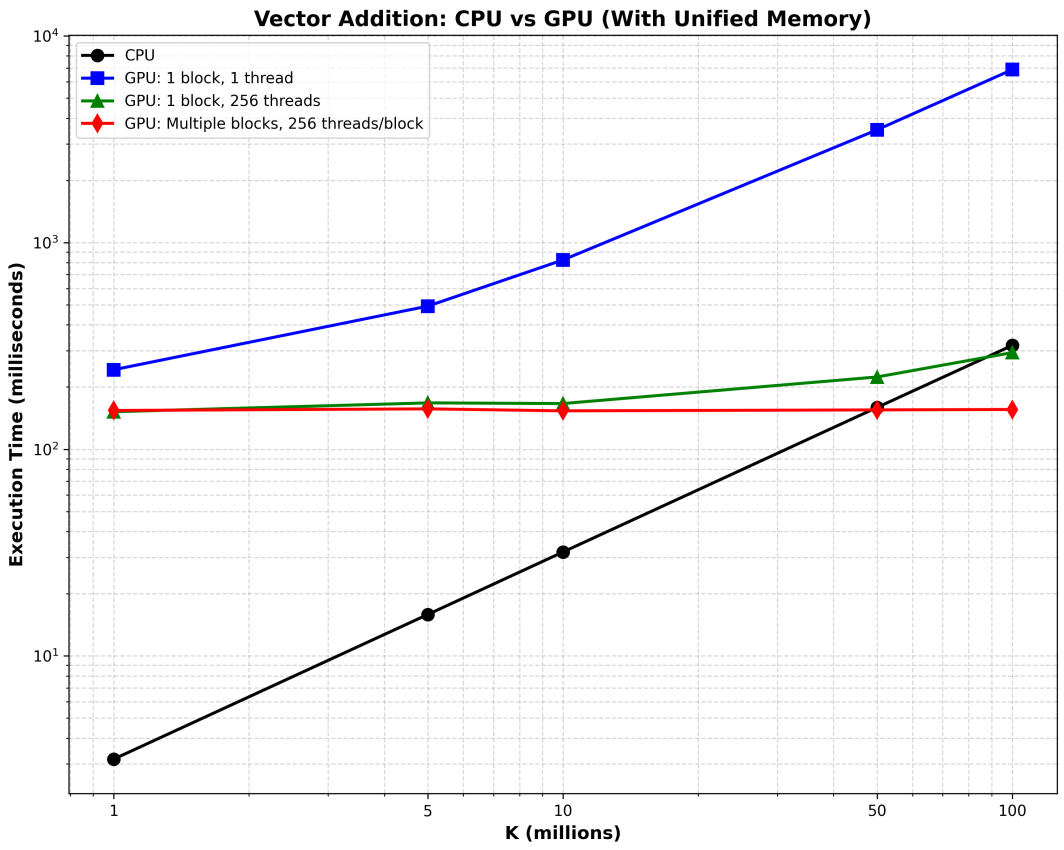


```
==========================================
SCENARIO 3: Multiple blocks, 256 threads/block
==========================================
K: 1 million, Grid size: 3907, Block size: 256
Unified Memory Allocation Time : 210.855 millisec
Execution Time                 : 0.019396 millisec
Total Time                     : 210.874 millisec
K: 5 million, Grid size: 19532, Block size: 256
Unified Memory Allocation Time : 152.475 millisec
Execution Time                 : 0.245634 millisec
Total Time                     : 152.72 millisec
K: 10 million, Grid size: 39063, Block size: 256
Unified Memory Allocation Time : 152.159 millisec
Execution Time                 : 0.464517 millisec
Total Time                     : 152.623 millisec
K: 50 million, Grid size: 195313, Block size: 256
Unified Memory Allocation Time : 153.289 millisec
Execution Time                 : 2.47663 millisec
Total Time                     : 155.766 millisec
K: 100 million, Grid size: 390625, Block size: 256
Unified Memory Allocation Time : 152.469 millisec
Execution Time                 : 5.01543 millisec
Total Time                     : 157.485 millisec
Singularity>
```

Q4



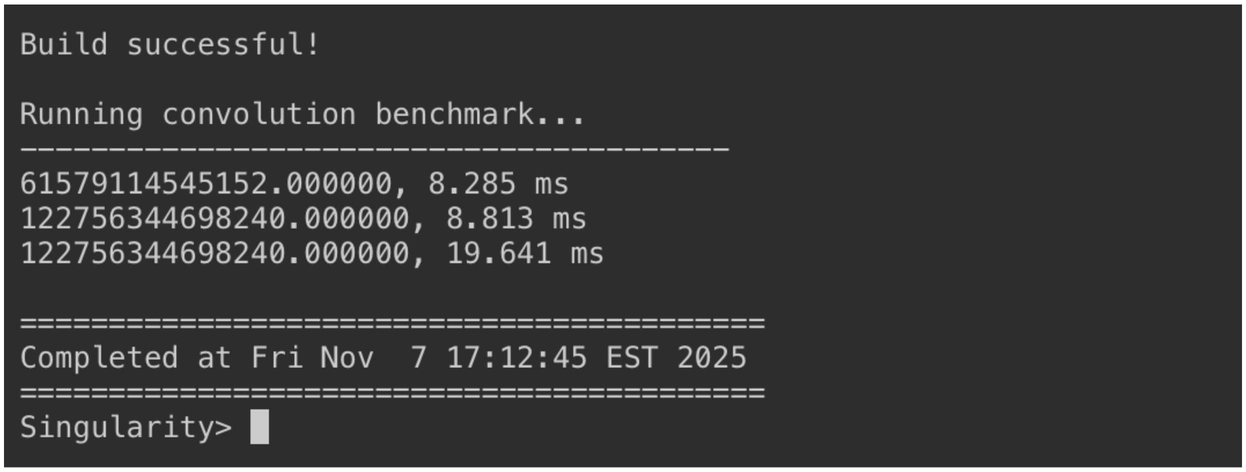
Vector Addition: CPU vs GPU (Without Unified Memory)
CPU
GPU: 1 block, 1 thread
GPU: 1 block, 256 threads
GPU: Multiple blocks, 256 threads/block
Execution Time (milliseconds)
K (millions)
10^1
10^2
10^3
10^4
1
5
10
50
100



Vector Addition: CPU vs GPU (With Unified Memory)
CPU
GPU: 1 block, 1 thread
GPU: 1 block, 256 threads
GPU: Multiple blocks, 256 threads/block
Execution Time (milliseconds)
K (millions)
10^1
10^2
10^3
10^4
1
5
10
50
100

Part C

Q1, Q2, Q3



```
Build successful!

Running convolution benchmark...
----------------------------------------
61579114545152.000000, 8.285 ms
122756344698240.000000, 8.813 ms
122756344698240.000000, 19.641 ms

==========================================
Completed at Fri Nov  7 17:12:45 EST 2025
==========================================
Singularity> 
```