CS 6501 - HW4 - CUDA

Huy Nguyen - mpc5ya

March 22, 2025

Question 7. 1 2 3 4 5

The code was run on GPU server 08, some variabilities may be different across GPU severs therefore different timing values may happen too.

Matrix Multiplication - matmul.cu

The following sections include results for four different configurations:

- **Small Matrix:** 10×10
- **Medium Matrix:** 128×128
- **Large Matrix:** 256×256
- **Extra Large Matrix:** 512×512

matmul.cu - using matrix size = 100 with width = 10

```
==479401== NVPROF is profiling process 479401, command: ./matmul_10x10
GPU takes 0.038206ms
CPU takes 0.003090ms
The matrix mul is right as both CPU and GPU matches
==479401== Profiling application: ./matmul_10x10
==479401== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   45.34%  314.49us       100  3.1440us  3.0720us  4.0330us  matrixMultiplication(int*,
                   35.83%  248.58us       200  1.2420us  1.0880us  1.6640us  [CUDA memcpy HtoD]
                   18.83%  130.63us       100  1.3060us  1.2480us  1.7600us  [CUDA memcpy DtoH]
      API calls:   85.40%  131.89ms         2  65.945ms  1.2550us  131.89ms  cudaEventCreate
                    5.51%  8.5122ms       300  28.373us  2.1350us  164.62us  cudaMalloc
                    4.91%  7.5863ms       300  25.287us  1.9370us  113.53us  cudaFree
                    1.71%  2.6429ms       300  8.8090us  4.6830us  24.254us  cudaMemcpy
                    1.16%  1.7851ms       404  4.4180us     394ns  206.74us  cuDeviceGetAttribute
                    0.56%  867.18us       100  8.6710us  7.5430us  31.593us  cudaLaunchKernel
                    0.33%  504.36us       200  2.5210us  1.8540us  10.120us  cudaEventRecord
                    0.29%  450.57us       100  4.5050us  2.8720us  14.039us  cudaEventSynchronize
                    0.08%  124.19us       100  1.2410us  1.0610us  8.3190us  cudaEventElapsedTime
                    0.03%  44.138us         4  11.034us  8.4560us  17.876us  cuDeviceGetName
                    0.01%  20.227us         4  5.0560us  3.0050us  10.405us  cuDeviceGetPCIBusId
                    0.00%  3.7670us         8     470ns     373ns  1.0050us  cuDeviceGet
                    0.00%  3.3350us         3  1.1110us     574ns  2.0560us  cuDeviceGetCount
                    0.00%  2.6750us         4     668ns     521ns     940ns  cuDeviceTotalMem
                    0.00%  2.2340us         2  1.1170us     577ns  1.6570us  cudaEventDestroy
```

```
                    0.00%  2.1830us         4     545ns     483ns     703ns  cuDeviceGetUuid
                    0.00%  1.1350us         1  1.1350us  1.1350us  1.1350us  cuModuleGetLoadingMode
```

matmul.cu - using matrix size = 16384 with width = 128

```
==479625== NVPROF is profiling process 479625, command: ./matmul_128x128
GPU takes 0.086644ms
CPU takes 6.351530ms
The matrix mul is right as both CPU and GPU matches
==479625== Profiling application: ./matmul_128x128
==479625== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   46.83%  1.3135ms       200  6.5670us  6.2080us  7.0710us  [CUDA memcpy HtoD]
                   31.84%  893.03us       100  8.9300us  8.7040us  9.6970us  matrixMultiplication(int*,
                   21.32%  598.01us       100  5.9800us  5.9200us  6.4310us  [CUDA memcpy DtoH]
      API calls:   82.30%  128.96ms         2  64.481ms  1.1460us  128.96ms  cudaEventCreate
                    5.49%  8.5986ms       300  28.661us  1.6530us  174.88us  cudaMalloc
                    5.03%  7.8763ms       300  26.254us  1.6970us  132.92us  cudaFree
                    4.69%  7.3430ms       300  24.476us  12.719us  65.439us  cudaMemcpy
                    1.14%  1.7829ms       404  4.4130us     395ns  219.20us  cuDeviceGetAttribute
                    0.63%  990.06us       100  9.9000us  8.5710us  31.765us  cudaLaunchKernel
                    0.33%  518.13us       200  2.5900us  1.8260us  19.073us  cudaEventRecord
                    0.28%  435.15us       100  4.3510us  2.2630us  6.3440us  cudaEventSynchronize
                    0.07%  111.81us       100  1.1180us  1.0290us  2.2480us  cudaEventElapsedTime
                    0.03%  42.902us         4  10.725us  8.2710us  17.093us  cuDeviceGetName
                    0.02%  24.906us         4  6.2260us  3.1900us  14.377us  cuDeviceGetPCIBusId
                    0.00%  4.1060us         2  2.0530us     556ns  3.5500us  cudaEventDestroy
                    0.00%  4.0200us         8     502ns     385ns  1.1230us  cuDeviceGet
                    0.00%  3.4340us         3  1.1440us     610ns  2.0780us  cuDeviceGetCount
                    0.00%  2.9970us         4     749ns     578ns  1.1540us  cuDeviceTotalMem
                    0.00%  2.1630us         4     540ns     490ns     623ns  cuDeviceGetUuid
                    0.00%     844ns         1     844ns     844ns     844ns  cuModuleGetLoadingMode
```

matmul.cu - using matrix size = 65536 with width = 256

```
Using matrix size = 65536 with width = 256
==479680== NVPROF is profiling process 479680, command: ./matmul_256x256
GPU takes 0.246796ms
CPU takes 50.248276ms
The matrix mul is right as both CPU and GPU matches
==479680== Profiling application: ./matmul_256x256
==479680== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   42.17%  4.8581ms       200  24.290us  24.031us  24.737us  [CUDA memcpy HtoD]
                   39.44%  4.5444ms       100  45.443us  44.928us  46.784us  matrixMultiplication(int*,
                   18.39%  2.1186ms       100  21.185us  21.119us  24.064us  [CUDA memcpy DtoH]
      API calls:   73.83%  127.83ms         2  63.915ms  1.0860us  127.83ms  cudaEventCreate
                   13.12%  22.719ms       300  75.731us  35.715us  170.28us  cudaMemcpy
                    5.34%  9.2383ms       300  30.794us  1.6560us  172.02us  cudaMalloc
                    5.30%  9.1790ms       300  30.596us  1.7000us  130.91us  cudaFree
                    1.02%  1.7633ms       404  4.3640us     401ns  204.63us  cuDeviceGetAttribute
                    0.69%  1.1957ms       100  11.957us  10.214us  33.958us  cudaLaunchKernel
                    0.34%  589.06us       200  2.9450us  2.1020us  13.234us  cudaEventRecord
                    0.24%  409.89us       100  4.0980us  1.6030us  6.6170us  cudaEventSynchronize
```

```
                    0.07%  120.12us       100  1.2010us  1.1060us  1.8310us  cudaEventElapsedTime
                    0.03%  47.803us         4  11.950us  8.6220us  21.470us  cuDeviceGetName
                    0.01%  22.651us         4  5.6620us  3.1610us  12.273us  cuDeviceGetPCIBusId
                    0.00%  5.8730us         2  2.9360us     884ns  4.9890us  cudaEventDestroy
                    0.00%  4.0070us         8     500ns     374ns  1.2420us  cuDeviceGet
                    0.00%  3.3740us         3  1.1240us     588ns  2.0940us  cuDeviceGetCount
                    0.00%  2.7860us         4     696ns     496ns     880ns  cuDeviceTotalMem
                    0.00%  2.1430us         4     535ns     470ns     690ns  cuDeviceGetUuid
                    0.00%  1.2540us         1  1.2540us  1.2540us  1.2540us  cuModuleGetLoadingMode
```

matmul.cu - using matrix size = 262144 with width = 512

```
GPU takes 0.953912ms
CPU takes 719.664917ms
The matrix mul is right as both CPU and GPU matches
==479736== Profiling application: ./matmul_512x512
==479736== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   54.57%  31.053ms       100  310.53us  309.25us  311.78us  matrixMultiplication(int*,
                   31.22%  17.768ms       200  88.841us  88.448us  97.504us  [CUDA memcpy HtoD]
                   14.21%  8.0856ms       100  80.855us  80.736us  88.256us  [CUDA memcpy DtoH]
      API calls:   48.68%  126.30ms         2  63.148ms  1.2380us  126.30ms  cudaEventCreate
                   35.94%  93.246ms       300  310.82us  113.81us  817.08us  cudaMemcpy
                    7.53%  19.534ms       300  65.114us  4.9990us  142.36us  cudaFree
                    6.14%  15.934ms       300  53.113us  2.0520us  157.30us  cudaMalloc
                    0.69%  1.7854ms       404  4.4190us     401ns  203.98us  cuDeviceGetAttribute
                    0.50%  1.2843ms       100  12.842us  11.565us  39.764us  cudaLaunchKernel
                    0.28%  718.78us       200  3.5930us  2.7570us  12.160us  cudaEventRecord
                    0.16%  410.41us       100  4.1040us  1.6890us  6.8740us  cudaEventSynchronize
                    0.05%  122.36us       100  1.2230us  1.0980us  1.8970us  cudaEventElapsedTime
                    0.02%  44.987us         4  11.246us  8.7690us  17.804us  cuDeviceGetName
                    0.01%  23.429us         4  5.8570us  3.0660us  12.993us  cuDeviceGetPCIBusId
                    0.00%  8.3520us         2  4.1760us     866ns  7.4860us  cudaEventDestroy
                    0.00%  4.3800us         8     547ns     375ns  1.6060us  cuDeviceGet
                    0.00%  3.4200us         3  1.1400us     550ns  2.1370us  cuDeviceGetCount
                    0.00%  2.5460us         4     636ns     490ns     903ns  cuDeviceTotalMem
                    0.00%  2.2060us         4     551ns     490ns     696ns  cuDeviceGetUuid
                    0.00%  1.1360us         1  1.1360us  1.1360us  1.1360us  cuModuleGetLoadingMode
```

Matrix Addition - matadd.cu

The following sections include results for four different configurations:

- **Small Matrix:** 64×64
- **Medium Matrix:** 128×128
- **Large Matrix:** 256×256
- **Extra Large Matrix:** 512×512

matadd.cu - using matrix of size 4096, with width = 64 and height = 64

```
==471354== NVPROF is profiling process 471354, command: ./matadd_64x64
```

```
GPU takes 0.106451ms
CPU takes 0.013960ms
The vector add is right as both CPU and GPU matches
==471354== Profiling application: ./matadd_64x64
==471354== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   45.54%  422.68us       200  2.1130us  2.0470us  2.4640us  [CUDA memcpy HtoD]
                   29.52%  274.02us       100  2.7400us  2.6870us  3.3920us  matrixAddition(int*, int*,
                   24.94%  231.45us       100  2.3140us  2.2400us  2.6560us  [CUDA memcpy DtoH]
      API calls:   78.93%  131.90ms         2  65.950ms  1.8230us  131.90ms  cudaEventCreate
                    7.30%  12.192ms       300  40.638us  2.0710us  218.63us  cudaMalloc
                    6.40%  10.689ms       300  35.630us  1.8730us  162.41us  cudaFree
                    2.78%  4.6474ms       300  15.491us  10.300us  37.492us  cudaMemcpy
                    2.66%  4.4455ms       100  44.454us  8.9240us  3.1319ms  cudaLaunchKernel
                    1.07%  1.7844ms       456  3.9130us     403ns  203.50us  cuDeviceGetAttribute
                    0.46%  770.86us       200  3.8540us  2.1810us  19.910us  cudaEventRecord
                    0.26%  439.72us       100  4.3970us  2.2060us  6.6790us  cudaEventSynchronize
                    0.10%  165.66us       100  1.6560us  1.0480us  4.4700us  cudaEventElapsedTime
                    0.03%  45.923us         4  11.480us  8.5190us  19.792us  cuDeviceGetName
                    0.01%  20.379us         4  5.0940us  3.2540us  9.9230us  cuDeviceGetPCIBusId
                    0.00%  4.4020us         8     550ns     381ns  1.3480us  cuDeviceGet
                    0.00%  3.5590us         3  1.1860us     584ns  2.2450us  cuDeviceGetCount
                    0.00%  2.9050us         4     726ns     630ns     964ns  cuDeviceTotalMem
                    0.00%  2.1580us         4     539ns     470ns     700ns  cuDeviceGetUuid
                    0.00%  2.1260us         2  1.0630us     522ns  1.6040us  cudaEventDestroy
                    0.00%  1.2400us         1  1.2400us  1.2400us  1.2400us  cuModuleGetLoadingMode
```

matadd.cu - using matrix of size 16384, with width = 128 and height = 128

```
==470638== NVPROF is profiling process 470638, command: ./matadd
GPU takes 0.904324ms
CPU takes 0.055880ms
The vector add is right as both CPU and GPU matches
==470638== Profiling application: ./matadd
==470638== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   59.97%  1.3272ms       200  6.6360us  6.2720us  7.1360us  [CUDA memcpy HtoD]
                   27.28%  603.74us       100  6.0370us  5.9520us  6.3690us  [CUDA memcpy DtoH]
                   12.75%  282.11us       100  2.8210us  2.7510us  3.6160us  matrixAddition(int*, int*,
      API calls:   51.14%  124.45ms         2  62.227ms  1.1640us  124.45ms  cudaEventCreate
                   33.00%  80.297ms       100  802.97us  10.356us  78.746ms  cudaLaunchKernel
                    5.75%  13.994ms       300  46.647us  1.8600us  212.29us  cudaMalloc
                    5.13%  12.492ms       300  41.639us  1.8440us  251.82us  cudaFree
                    3.57%  8.6804ms       300  28.934us  16.241us  102.83us  cudaMemcpy
                    0.74%  1.8026ms       456  3.9520us     401ns  205.94us  cuDeviceGetAttribute
                    0.36%  887.46us       200  4.4370us  2.2080us  16.973us  cudaEventRecord
                    0.18%  444.91us       100  4.4490us  2.0000us  9.6490us  cudaEventSynchronize
                    0.08%  198.94us       100  1.9890us  1.0860us  7.2830us  cudaEventElapsedTime
                    0.02%  45.220us         4  11.305us  8.3260us  19.570us  cuDeviceGetName
                    0.02%  37.444us         4  9.3610us  3.2970us  26.977us  cuDeviceGetPCIBusId
                    0.00%  4.4090us         8     551ns     382ns  1.5410us  cuDeviceGet
                    0.00%  3.5640us         3  1.1880us     538ns  2.2000us  cuDeviceGetCount
                    0.00%  3.4960us         2  1.7480us     689ns  2.8070us  cudaEventDestroy
                    0.00%  2.4570us         4     614ns     468ns     989ns  cuDeviceTotalMem
```

```
                    0.00%  1.9680us         4     492ns     443ns     605ns  cuDeviceGetUuid
                    0.00%  1.0580us         1  1.0580us  1.0580us  1.0580us  cuModuleGetLoadingMode
```

matadd.cu - using matrix of size 65536, with width = 256 and height = 256

```
==470723== NVPROF is profiling process 470723, command: ./matadd_256x256
GPU takes 0.237096ms
CPU takes 0.209650ms
The vector add is right as both CPU and GPU matches
==470723== Profiling application: ./matadd_256x256
==470723== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   66.39%  4.8514ms       200  24.257us  23.968us  24.544us  [CUDA memcpy HtoD]
                   29.05%  2.1230ms       100  21.229us  21.120us  27.327us  [CUDA memcpy DtoH]
                    4.56%  333.48us       100  3.3340us  3.2000us  11.297us  matrixAddition(int*, int*,
      API calls:   73.86%  121.70ms         2  60.851ms     872ns  121.70ms  cudaEventCreate
                   10.89%  17.943ms       300  59.808us  33.952us  131.91us  cudaMemcpy
                    5.49%  9.0387ms       300  30.129us  1.7340us  161.87us  cudaMalloc
                    5.34%  8.7969ms       300  29.322us  1.7500us  129.82us  cudaFree
                    2.60%  4.2915ms       100  42.915us  9.9900us  3.1810ms  cudaLaunchKernel
                    1.09%  1.7893ms       456  3.9230us     401ns  204.79us  cuDeviceGetAttribute
                    0.36%  591.42us       200  2.9570us  2.0950us  18.803us  cudaEventRecord
                    0.25%  414.36us       100  4.1430us  1.8060us  6.5360us  cudaEventSynchronize
                    0.07%  114.32us       100  1.1430us  1.0120us  3.4360us  cudaEventElapsedTime
                    0.03%  43.324us         4  10.831us  8.5450us  16.896us  cuDeviceGetName
                    0.01%  22.385us         4  5.5960us  3.1460us  12.683us  cuDeviceGetPCIBusId
                    0.00%  4.4340us         2  2.2170us     705ns  3.7290us  cudaEventDestroy
                    0.00%  4.3540us         8     544ns     379ns  1.4600us  cuDeviceGet
                    0.00%  3.7450us         3  1.2480us     576ns  2.3840us  cuDeviceGetCount
                    0.00%  2.5910us         4     647ns     476ns     976ns  cuDeviceTotalMem
                    0.00%  2.2930us         4     573ns     504ns     739ns  cuDeviceGetUuid
                    0.00%  1.1650us         1  1.1650us  1.1650us  1.1650us  cuModuleGetLoadingMode
```

matadd.cu - using matrix of size 262144, with width = 512 and height = 512

```
==470795== NVPROF is profiling process 470795, command: ./matadd_512x512
GPU takes 0.715899ms
CPU takes 0.829780ms
The vector add is right as both CPU and GPU matches
==470795== Profiling application: ./matadd_512x512
==470795== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   67.33%  17.781ms       200  88.902us  88.480us  97.439us  [CUDA memcpy HtoD]
                   30.64%  8.0904ms       100  80.904us  80.769us  88.288us  [CUDA memcpy DtoH]
                    2.03%  535.49us       100  5.3540us  5.1840us  6.2080us  matrixAddition(int*, int*,
      API calls:   52.06%  120.28ms         2  60.141ms  1.0960us  120.28ms  cudaEventCreate
                   28.50%  65.856ms       300  219.52us  114.68us  817.45us  cudaMemcpy
                    8.91%  20.577ms       300  68.589us  4.9560us  153.45us  cudaFree
                    7.35%  16.987ms       300  56.622us  2.0290us  168.40us  cudaMalloc
                    1.76%  4.0742ms       100  40.742us  11.309us  2.6831ms  cudaLaunchKernel
                    0.79%  1.8149ms       456  3.9800us     404ns  209.43us  cuDeviceGetAttribute
                    0.36%  829.27us       200  4.1460us  2.7720us  21.345us  cudaEventRecord
                    0.17%  403.94us       100  4.0390us  1.6260us  7.4900us  cudaEventSynchronize
```

```
                    0.06%  142.31us       100  1.4230us  1.0980us  4.1190us  cudaEventElapsedTime
                    0.02%  42.814us         4  10.703us  8.5320us  16.824us  cuDeviceGetName
                    0.01%  23.027us         4  5.7560us  2.5120us  13.375us  cuDeviceGetPCIBusId
                    0.00%  5.2050us         2  2.6020us     734ns  4.4710us  cudaEventDestroy
                    0.00%  4.3240us         8     540ns     388ns  1.4560us  cuDeviceGet
                    0.00%  3.7050us         3  1.2350us     693ns  2.2760us  cuDeviceGetCount
                    0.00%  2.5710us         4     642ns     489ns     864ns  cuDeviceTotalMem
                    0.00%  2.2010us         4     550ns     475ns     677ns  cuDeviceGetUuid
                    0.00%  1.1900us         1  1.1900us  1.1900us  1.1900us  cuModuleGetLoadingMode
```

Parallel Sum - parallelSum.cu

The following sections include results for four different configurations:

- **Small Array:** $width = 2^{13} = 8192$
- **Medium Array:** $width = 2^{14} = 16384$
- **Large Array:** $width = 2^{15} = 32768$
- **Extra Large Array:** $width = 2^{16} = 65536$

parallelSum.cu - using array size of 8192

```
Using array size of 8192
==477706== NVPROF is profiling process 477706, command: ./parallelSum_2x13
GPU with parallel sum takes 0.041456ms
CPU with normal sum takes 0.018360ms
The parallelSum was running correctly on the GPU as results on both CPU and GPU match
CPU Sum = 36725
GPU Sum = 36725
==477706== Profiling application: ./parallelSum_2x13
==477706== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   57.06%  687.24us       200  3.4360us  3.2640us  4.1910us  parallelSum(int*, int*, in
                   32.24%  388.32us       100  3.8830us  3.3910us  4.2560us  [CUDA memcpy HtoD]
                   10.70%  128.83us       100  1.2880us  1.2470us  1.7920us  [CUDA memcpy DtoH]
      API calls:   84.49%  122.17ms         2  61.086ms  3.1280us  122.17ms  cudaEventCreate
                    5.73%  8.2896ms       300  27.632us  2.0700us  161.54us  cudaMalloc
                    5.14%  7.4345ms       300  24.781us  1.9270us  104.35us  cudaFree
                    1.80%  2.6042ms       200  13.020us  11.311us  30.879us  cudaMemcpy
                    1.22%  1.7653ms       404  4.3690us     404ns  203.23us  cuDeviceGetAttribute
                    0.85%  1.2242ms       200  6.1210us  3.4250us  42.481us  cudaLaunchKernel
                    0.34%  485.11us       200  2.4250us  1.8360us  12.972us  cudaEventRecord
                    0.31%  446.49us       100  4.4640us  2.8800us  6.1690us  cudaEventSynchronize
                    0.07%  99.394us       100     993ns     891ns  2.6130us  cudaEventElapsedTime
                    0.03%  46.232us         4  11.558us  8.4700us  19.890us  cuDeviceGetName
                    0.02%  22.981us         4  5.7450us  2.8510us  13.337us  cuDeviceGetPCIBusId
                    0.00%  3.8770us         8     484ns     371ns  1.0620us  cuDeviceGet
                    0.00%  3.3420us         3  1.1140us     555ns  2.0920us  cuDeviceGetCount
                    0.00%  2.7970us         4     699ns     552ns     886ns  cuDeviceTotalMem
                    0.00%  2.1850us         4     546ns     466ns     720ns  cuDeviceGetUuid
                    0.00%  1.0770us         1  1.0770us  1.0770us  1.0770us  cuModuleGetLoadingMode
```

parallelSum.cu - using array size of 16384

```
==477399== NVPROF is profiling process 477399, command: ./parallelSum_2x14
GPU with parallel sum takes 0.046595ms
CPU with normal sum takes 0.037290ms
The parallelSum was running correctly on the GPU as results on both CPU and GPU match
CPU Sum = 73975
GPU Sum = 73975
==477399== Profiling application: ./parallelSum_2x14
==477399== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   46.90%  702.17us       200  3.5100us  3.2950us  4.4480us  parallelSum(int*, int*, in
                   44.40%  664.77us       100  6.6470us  6.3360us  7.2640us  [CUDA memcpy HtoD]
                    8.71%  130.34us       100  1.3030us  1.2480us  1.7910us  [CUDA memcpy DtoH]
      API calls:   84.43%  126.37ms         2  63.187ms  3.5150us  126.37ms  cudaEventCreate
                    5.64%  8.4371ms       300  28.123us  2.1400us  154.96us  cudaMalloc
                    5.07%  7.5963ms       300  25.320us  1.9560us  116.32us  cudaFree
                    2.13%  3.1821ms       200  15.910us  11.727us  32.101us  cudaMemcpy
                    1.21%  1.8139ms       404  4.4890us     403ns  240.58us  cuDeviceGetAttribute
                    0.78%  1.1608ms       200  5.8040us  3.5360us  29.006us  cudaLaunchKernel
                    0.32%  482.21us       200  2.4110us  1.7250us  16.439us  cudaEventRecord
                    0.30%  445.69us       100  4.4560us  2.7140us  6.1090us  cudaEventSynchronize
                    0.08%  115.25us       100  1.1520us     965ns  7.7390us  cudaEventElapsedTime
                    0.03%  41.391us         4  10.347us  7.7800us  16.443us  cuDeviceGetName
                    0.01%  18.819us         4  4.7040us  2.1680us  10.218us  cuDeviceGetPCIBusId
                    0.00%  4.1510us         8     518ns     382ns  1.3040us  cuDeviceGet
                    0.00%  3.5410us         3  1.1800us     612ns  2.2590us  cuDeviceGetCount
                    0.00%  2.6680us         4     667ns     560ns     941ns  cuDeviceTotalMem
                    0.00%  2.4380us         4     609ns     500ns     737ns  cuDeviceGetUuid
                    0.00%     809ns         1     809ns     809ns     809ns  cuModuleGetLoadingMode
```

parallelSum.cu - using array size of 32768

```
==477262== NVPROF is profiling process 477262, command: ./parallelSum_2x15
GPU with parallel sum takes 0.064286ms
CPU with normal sum takes 0.074260ms
The parallelSum was running correctly on the GPU as results on both CPU and GPU match
CPU Sum = 147590
GPU Sum = 147590
==477262== Profiling application: ./parallelSum_2x15
==477262== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   61.03%  1.3507ms       100  13.506us  13.248us  13.953us  [CUDA memcpy HtoD]
                   33.10%  732.60us       200  3.6630us  3.2640us  5.3120us  parallelSum(int*, int*, in
                    5.87%  130.02us       100  1.3000us  1.2480us  1.7280us  [CUDA memcpy DtoH]
      API calls:   83.12%  123.09ms         2  61.545ms  1.2340us  123.09ms  cudaEventCreate
                    5.59%  8.2823ms       300  27.607us  1.7140us  172.12us  cudaMalloc
                    5.13%  7.5914ms       300  25.304us  1.6030us  118.14us  cudaFree
                    3.27%  4.8407ms       200  24.203us  11.380us  39.302us  cudaMemcpy
                    1.27%  1.8872ms       404  4.6710us     409ns  280.31us  cuDeviceGetAttribute
                    0.84%  1.2471ms       200  6.2350us  3.6350us  29.983us  cudaLaunchKernel
                    0.34%  508.82us       200  2.5440us  1.8690us  13.739us  cudaEventRecord
                    0.30%  441.05us       100  4.4100us  2.6450us  6.1350us  cudaEventSynchronize
                    0.08%  113.33us       100  1.1330us  1.0100us  2.8880us  cudaEventElapsedTime
                    0.03%  44.426us         4  11.106us  8.3670us  18.304us  cuDeviceGetName
```

```
                    0.01%  19.177us         4  4.7940us  3.1470us  9.2250us  cuDeviceGetPCIBusId
                    0.00%  4.3450us         8     543ns     390ns  1.3310us  cuDeviceGet
                    0.00%  3.3430us         3  1.1140us     528ns  2.1280us  cuDeviceGetCount
                    0.00%  2.6570us         4     664ns     490ns     880ns  cuDeviceTotalMem
                    0.00%  2.4060us         4     601ns     470ns     755ns  cuDeviceGetUuid
                    0.00%  1.1680us         1  1.1680us  1.1680us  1.1680us  cuModuleGetLoadingMode
```

parallelSum.cu - using array size of 65536

```
==477061== NVPROF is profiling process 477061, command: ./parallelSum_2x16
GPU with parallel sum takes 0.093416ms
CPU with normal sum takes 0.149400ms
The parallelSum was running correctly on the GPU as results on both CPU and GPU match
CPU Sum = 294625
GPU Sum = 294625
==477061== Profiling application: ./parallelSum_2x16
==477061== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   71.19%  2.4375ms       100  24.374us  24.160us  24.960us  [CUDA memcpy HtoD]
                   24.98%  855.14us       200  4.2750us  3.2960us  5.9530us  parallelSum(int*, int*, in
                    3.83%  131.27us       100  1.3120us  1.2480us  1.7600us  [CUDA memcpy DtoH]
      API calls:   81.68%  128.91ms         2  64.457ms  3.6300us  128.91ms  cudaEventCreate
                    5.59%  8.8166ms       300  29.388us  1.6350us  175.29us  cudaMalloc
                    5.17%  8.1596ms       300  27.198us  1.6050us  129.67us  cudaFree
                    4.78%  7.5489ms       200  37.744us  18.723us  67.000us  cudaMemcpy
                    1.13%  1.7803ms       404  4.4060us     400ns  203.71us  cuDeviceGetAttribute
                    0.90%  1.4193ms       200  7.0960us  3.7500us  31.260us  cudaLaunchKernel
                    0.36%  564.01us       200  2.8200us  1.9800us  19.272us  cudaEventRecord
                    0.27%  429.77us       100  4.2970us  1.5360us  6.3680us  cudaEventSynchronize
                    0.07%  116.97us       100  1.1690us  1.0600us  3.2320us  cudaEventElapsedTime
                    0.03%  45.138us         4  11.284us  8.3340us  18.744us  cuDeviceGetName
                    0.02%  25.237us         4  6.3090us  3.2150us  14.780us  cuDeviceGetPCIBusId
                    0.00%  4.0670us         8     508ns     392ns  1.1880us  cuDeviceGet
                    0.00%  3.3780us         3  1.1260us     524ns  2.2240us  cuDeviceGetCount
                    0.00%  2.7310us         4     682ns     539ns  1.0800us  cuDeviceTotalMem
                    0.00%  2.2380us         4     559ns     449ns     700ns  cuDeviceGetUuid
                    0.00%  1.2080us         1  1.2080us  1.2080us  1.2080us  cuModuleGetLoadingMode
```

Question 7.5: Timings over 100 iterations for each CUDA programs

Matrix Multiplication (`matmul.cu`)

We tested four matrix sizes: 10×10, 128×128, 256×256, and 512×512. Table 1 now includes:

- **CPU Time (ms):** From the code output (timed on the host).
- **GPU Time (ms, from code):** From the code's `cudaEvent` timing (usually just one main launch).
- **Kernel Execution Time (ms, from NVPROF):** Sum of all kernel calls of `matrixMultiplication()` as reported by NVPROF.
- **Data Movement (ms):** Sum of HtoD + DtoH times from NVPROF.

Matrix Size	CPU Time (ms)	GPU Time (ms)	Kernel Execution Time (ms)	Data Movement (ms)
10×10	0.00309	0.03821	0.31449	0.37921
128×128	6.35153	0.08664	0.89303	1.91150
256×256	50.24828	0.24680	4.54440	6.97670
512×512	719.66492	0.95391	31.05300	25.85360

Table 1: Extended timings for `matmul.cu`. "Kernel Execution Time" is the total `matrixMultiplication()` time from NVPROF.

1. **For what input size does the GPU outperform the CPU?**
   By inspecting the "CPU Time (ms)" versus "GPU Time (ms)" columns in Table 1, we see that:
   - At 10×10, CPU time (0.00309 ms) is smaller than GPU time (0.03821 ms), which means that the CPU is faster when matrix size is smaller.
   - At 128×128 and beyond, the GPU time becomes significantly faster than the CPU time (e.g., 0.08664 ms vs. 6.35153 ms at 128×128), which means that the GPU parallelism comes in effectively for larger matrix size.

   Therefore, the GPU *starts outperforming* the CPU beginning at $\boxed{128 \times 128}$, or any large matrix size.

2. **For what input size is data movement time less than the kernel execution time on the GPU?**
   Compare the "Kernel Execution (ms)" column to the "Data Movement (ms)" column:
   - 10×10: Data movement = 0.37921 ms > Kernel = 0.31449 ms
   - 128×128: Data movement = 1.91150 ms > Kernel = 0.89303 ms
   - 256×256: Data movement = 6.97670 ms > Kernel = 4.54440 ms
   - 512×512: Data movement = 25.85360 ms < Kernel = 31.05300 ms

   Only at $\boxed{512 \times 512}$ do we see that data-transfer overhead (25.85360 ms) is *less* than the total kernel execution time (31.05300 ms). This means that for large matrix size, it may take longer to actually do the computation even with GPU, and data transfer is faster.

Matrix Addition (`matadd.cu`)

Similarly, for matrix addition we tested 64×64, 128×128, 256×256, and 512×512. Table 2 adds the kernel-execution column for `matrixAddition()`:

Matrix Size	CPU Time (ms)	GPU Time (ms)	Kernel Execution Time (ms)	Data Movement (ms)
64×64	0.01396	0.10645	0.27402	0.65413
128×128	0.05588	0.90432	0.28211	1.93094
256×256	0.20965	0.23710	0.33348	6.97440
512×512	0.82978	0.71590	0.53549	25.87140

Table 2: Extended timings for `matadd.cu`. "Kernel Execution Time" is the total `matrixAddition()` time from NVPROF.

1. **For what input size does the GPU outperform the CPU?**
   By inspecting the "CPU Time (ms)" vs. "GPU Time (ms)" columns in Table 2, we see:

- For 64×64 and 128×128, the CPU is faster (e.g., 0.014 ms vs. 0.106 ms at 64×64, and 0.056 ms vs. 0.904 ms at 128×128).
- At 256×256, the CPU takes ≈ 0.210ms, and the GPU takes ≈ 0.237ms, so the CPU is still slightly faster (though the difference is small).
- At 512×512, the GPU time (0.716ms) is lower than the CPU time (0.830ms), so the GPU *finally* outperforms the CPU.

Thus, the GPU starts to *consistently outperform* the CPU beginning at $\boxed{512 \times 512}$ and for any larger matrix size. Since vector addition does not take a lot of computing power, the CPU may outperforms GPU at small matrix size due to overhead of kernel launching and data transfer. However, when the matrix size got larger, the GPU can still outperforms CPU, but at a later size compared to matrix multiplication since matrix multiplication is more computational heavy, therefore the GPU comes in place faster.

2. **For what input size is the data movement time less than the kernel execution time on the GPU?**
Compare the "Kernel Execution (ms)" column to the "Data Movement (ms)" column for each size:

   - 64×64: Data movement = 0.65413ms, kernel = 0.27402ms (Data > Kernel)
   - 128×128: Data movement = 1.93094ms, kernel = 0.28211ms (Data > Kernel)
   - 256×256: Data movement = 6.97440ms, kernel = 0.33348ms (Data > Kernel)
   - 512×512: Data movement = 25.87140ms, kernel = 0.53549ms (Data $\gg$ Kernel)

   In all cases, the total data-transfer time exceeds the kernel's execution time. Hence, for these four matrix sizes, there is *no* instance where Data Movement < Kernel Execution.

   Since Matrix addition is a relatively simple (low-arithmetic) operation: each element in the matrix requires just one addition. Consequently, the amount of time spent on actual GPU computation remains small, especially compared to the overhead of transferring matrices back and forth between the host (CPU) and device (GPU).

Parallel Sum (`parallelSum.cu`)

Finally, we tested parallel sum on arrays of size 8192, 16384, 32768, and 65536. Table 3 includes both the CPU/GPU timings from the code and the total `parallelSum()` time from NVPROF. (In some cases, we approximate the kernel time for the largest size based on partial logs.)

Array Size	CPU Time (ms)	GPU Time (ms)	Kernel Execution Time (ms)	Data Movement (ms)
8192	0.01836	0.04146	0.68724	0.51715
16384	0.03729	0.04660	0.70217	0.79510
32768	0.07426	0.06429	0.73260	1.48070
65536	0.14940	0.09342	≈ 0.76	2.56880

Table 3: Extended timings for `parallelSum.cu`. "Kernel Execution Time" is the total `parallelSum()` time (sum of all calls) from NVPROF. The 65536-kernel time is an approximation based on observed scaling.

1. **For what input size does the GPU outperform the CPU?**
From Table 3, comparing "CPU Time (ms)" vs. "GPU Time (ms)":

   - 8192: CPU = 0.01836 ms, GPU = 0.04146 ms $\rightarrow$ CPU faster
   - 16384: CPU = 0.03729 ms, GPU = 0.04660 ms $\rightarrow$ CPU still faster
   - 32768: CPU = 0.07426 ms, GPU = 0.06429 ms $\rightarrow$ GPU becomes faster

- 65536: CPU = 0.14940 ms, GPU = 0.09342 ms → GPU remains faster

Thus, the GPU *starts outperforming* the CPU beginning at $\boxed{32768}$ elements.

2. **For what input size is data movement time less than the kernel execution time on the GPU?**
Compare the "Kernel Execution (ms)" to the "Data Movement (ms)" columns:

   - 8192: Data movement = 0.51715 ms, kernel = 0.68724 ms → $0.51715 < 0.68724$
   - 16384: Data movement = 0.79510 ms, kernel = 0.70217 ms → $0.79510 > 0.70217$
   - 32768: Data movement = 1.48070 ms, kernel = 0.73260 ms → $1.48070 > 0.73260$
   - 65536: Data movement = 2.56880 ms, kernel ≈ 0.76 ms → $2.56880 > 0.76$

   The only case where Data Movement < Kernel Execution is $\boxed{8192}$ elements.

Conclusion

- **Matrix Multiplication (`matmul.cu`):**
  - The GPU starts outperforming the CPU at $\mathbf{128 \times 128}$. This is because matrix multiplication is a computationally expensive operation ($\mathcal{O}(n^3)$), making the GPU's parallelism highly effective as the matrix size grows.
  - Data movement time becomes smaller than kernel execution time only at $\mathbf{512 \times 512}$. At smaller sizes, the overhead of transferring matrices between CPU and GPU dominates the total execution time.
- **Matrix Addition (`matadd.cu`):**
  - The GPU starts outperforming the CPU at $\mathbf{512 \times 512}$. Unlike matrix multiplication, addition is a low-computation operation ($\mathcal{O}(n^2)$), so the CPU handles smaller matrices efficiently, and GPU overhead (kernel launch and memory transfer) delays its advantage until larger sizes.
  - Data movement time *always* exceeds kernel execution time. Since addition requires only one operation per element, the actual computation time remains small compared to the cost of transferring the matrix data.
- **Parallel Sum (`parallelSum.cu`):**
  - The GPU overtakes the CPU at **32768** elements. Parallel reduction algorithms benefit from the GPU's ability to process multiple additions simultaneously, but at smaller sizes, the CPU's sequential execution remains competitive.
  - Data movement time is only less than kernel execution time at **8192** elements. As the array size increases, the cost of transferring data grows significantly, making it the dominant factor in execution time.

Question 8: Kernel Launch Differences in `matmul.cu`, `matadd.cu`, and `parallelSum.cu`

The kernel launch configurations differ between `matmul.cu`, `matadd.cu`, and `parallelSum.cu` due to the nature of the computations they perform as the **parallelSum use Shared Memory.**

- **Matrix Multiplication and Addition:** In `matmul.cu` and `matadd.cu`, the computation is structured in a two-dimensional grid, as each thread processes an element in a 2D matrix. The kernel launch typically follows:

  ```
  <<<gridDim, blockDim>>>
  ```

  where `gridDim` and `blockDim` are two-dimensional configurations that match the matrix structure.

- **Parallel Reduction in `parallelSum.cu`:** The kernel launch for `parallelSum.cu` includes three parameters:

  `<<<gridDim, blockDim, sharedMemSize>>>`

  The third parameter, `sharedMemSize`, is used to allocate shared memory dynamically. This is necessary for parallel reduction, where threads within a block share data to iteratively compute the sum in a hierarchical manner. Shared memory reduces global memory accesses and improves efficiency.

Question 9: Purpose of `__syncthreads()` in `parallelSum.cu`

The `__syncthreads()` function is a barrier synchronization primitive used in CUDA to ensure that all threads within a block reach the same execution point before proceeding. In `parallelSum.cu`, it is good for:

- The sync is place after loading data into shared mem to prevent race conditions by ensuring that all threads complete their reads/writes to shared memory before continuing.
- One more sync is placed after reduction to ensure partial sums to be correctly accumulated for the current stride value first, then after all the results were completed for that reduction, only that the next iterations of stride begin, since the next iterations of strides need the previous stride value results, therefore we need syncthread().

Without `__syncthreads()`, some threads might read incomplete or inconsistent values, leading to incorrect results in the parallel sum computation.

Question 10: Shared Mem vs Regular Matrix Mul

a. Compare the runtime difference of *matmul_sharedmem.cu* with *matmul.cu* (without shared memory).

Using SMALL matrix size = 4096 with width = 64 and Using shared memory tile size = 32 x 32

CPU takes 0.859600ms
GPU Regular takes 0.033994ms
GPU Shared Memory takes 0.042090ms
Speedup (Shared vs Regular): 0.81x
Speedup (Shared vs CPU): 20.42x
The matrix mul is right as both CPU and GPU matches.

```
==489254== NVPROF is profiling process 489254, command: ./matmul_sharedmem_64x64
==489254== Profiling application: ./matmul_sharedmem_64x64
==489254== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   32.09%  87.261us        11  7.9320us  7.7750us  8.7360us  matrixMultiplicationShared
                   31.55%  85.795us        42  2.0420us  1.9520us  2.3040us  [CUDA memcpy HtoD]
                   18.21%  49.535us        21  2.3580us  2.2720us  2.6560us  [CUDA memcpy DtoH]
                   18.16%  49.377us        10  4.9370us  4.8640us  5.2160us  matrixMultiplication(int*,
      API calls:   94.39%  137.83ms         2  68.914ms  3.5630us  137.83ms  cudaEventCreate
                    1.80%  2.6285ms        63  41.721us  2.4750us  326.12us  cudaMalloc
                    1.49%  2.1792ms        63  34.590us  2.4100us  136.52us  cudaFree
                    1.21%  1.7695ms       404  4.3800us     406ns  203.31us  cuDeviceGetAttribute
                    0.68%  993.44us        63  15.768us  9.8480us  49.310us  cudaMemcpy
                    0.20%  292.58us        21  13.932us  11.442us  37.146us  cudaLaunchKernel
                    0.09%  134.06us        40  3.3510us  2.6540us  7.3470us  cudaEventRecord
                    0.06%  87.454us        20  4.3720us  2.0100us  4.8280us  cudaEventSynchronize
```

```
                    0.03%  43.218us         4  10.804us  8.3950us  17.473us  cuDeviceGetName
                    0.02%  32.152us        20  1.6070us  1.4830us  2.5160us  cudaEventElapsedTime
                    0.01%  20.695us         4  5.1730us  3.1200us  10.199us  cuDeviceGetPCIBusId
                    0.00%  4.1580us         8     519ns     370ns  1.4260us  cuDeviceGet
                    0.00%  3.5590us         3  1.1860us     636ns  2.1930us  cuDeviceGetCount
                    0.00%  3.0110us         4     752ns     540ns  1.1600us  cuDeviceTotalMem
                    0.00%  2.4560us         2  1.2280us     654ns  1.8020us  cudaEventDestroy
                    0.00%  2.2460us         4     561ns     498ns     725ns  cuDeviceGetUuid
                    0.00%  1.1150us         1  1.1150us  1.1150us  1.1150us  cuModuleGetLoadingMode
```

Using LARGE matrix size = 262144 with width = 512 and Using shared memory tile size = 32 x 32

===== Performance Comparison =====
CPU takes 702.954468ms
GPU Regular takes 0.542010ms
GPU Shared Memory takes 0.445958ms
Speedup (Shared vs Regular): 1.22x
Speedup (Shared vs CPU): 1576.28x
The matrix mul is right as both CPU and GPU matches

```
==489421== Profiling application: ./matmul_sharedmem_512x512
==489421== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   35.31%  3.7307ms        42  88.827us  88.512us  91.264us  [CUDA memcpy HtoD]
                   29.38%  3.1040ms        10  310.40us  309.86us  310.82us  matrixMultiplication(int*,
                   19.17%  2.0257ms        11  184.16us  180.45us  185.18us  matrixMultiplicationShared
                   16.13%  1.7043ms        21  81.156us  80.768us  88.256us  [CUDA memcpy DtoH]
      API calls:   80.43%  123.68ms         2  61.841ms  3.0760us  123.68ms  cudaEventCreate
                   12.59%  19.366ms        63  307.39us  126.07us  688.81us  cudaMemcpy
                    2.85%  4.3833ms        63  69.576us  5.5230us  136.54us  cudaFree
                    2.51%  3.8539ms        63  61.172us  2.1540us  293.11us  cudaMalloc
                    1.16%  1.7761ms       404  4.3960us     403ns  207.87us  cuDeviceGetAttribute
                    0.21%  315.39us        21  15.018us  11.974us  34.312us  cudaLaunchKernel
                    0.10%  155.63us        40  3.8900us  2.7470us  14.342us  cudaEventRecord
                    0.06%  88.826us         4  22.206us  8.3820us  62.836us  cuDeviceGetName
                    0.05%  83.948us        20  4.1970us  1.8400us  4.6810us  cudaEventSynchronize
                    0.02%  29.092us        20  1.4540us  1.1840us  2.2670us  cudaEventElapsedTime
                    0.01%  18.140us         4  4.5350us  3.0240us  7.7420us  cuDeviceGetPCIBusId
                    0.00%  4.7780us         8     597ns     404ns  1.2800us  cuDeviceGet
                    0.00%  4.0440us         3  1.3480us     610ns  2.2550us  cuDeviceGetCount
                    0.00%  3.6970us         2  1.8480us     544ns  3.1530us  cudaEventDestroy
                    0.00%  2.8120us         4     703ns     540ns  1.0310us  cuDeviceTotalMem
                    0.00%  2.5290us         1  2.5290us  2.5290us  2.5290us  cuModuleGetLoadingMode
                    0.00%  2.3380us         4     584ns     496ns     824ns  cuDeviceGetUuid
```

In the case of large matrix multiplication (size = 262,144 with width = 512), using shared memory significantly improves performance over regular global memory access. The shared memory implementation achieves a runtime of 0.445958ms, compared to 0.542010ms for the regular GPU version—resulting in a 1.22× speedup. This performance gain is due to the ability of shared memory to reduce redundant global memory accesses by enabling data reuse within thread blocks, which becomes increasingly beneficial as matrix size grows and memory bandwidth becomes a bottleneck.

In contrast, for small matrices (size = 4,096 with width = 64), using shared memory actually leads to slightly worse performance, with the shared version taking 0.042090ms versus 0.033994ms for the regular

GPU method. Here, the overhead of managing shared memory and synchronizing threads outweighs its benefits, as the memory access pattern is already efficient due to GPU caching and the smaller working set size. This highlights that shared memory is more effective for larger workloads, while for smaller matrices, the simpler regular global memory approach can be more optimal.

Question 10.b: Using fixed matrix size of 512x512, finding the optimal tile size

Using shared memory tile size = 4 x 4

```
Using matrix size = 262144 with width = 512
Using shared memory tile size = 4 x 4
==490949== NVPROF is profiling process 490949, command: ./matmul_sharedmem_512x512_4

===== Performance Comparison =====
CPU takes 722.277832ms
GPU Regular takes 0.575514ms
GPU Shared Memory takes 1.342522ms
Speedup (Shared vs Regular): 0.43x
Speedup (Shared vs CPU): 538.00x
The matrix mul is right as both CPU and GPU matches
==490949== Profiling application: ./matmul_sharedmem_512x512_4
==490949== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   55.95%  10.745ms        11  976.80us  972.22us  987.36us  matrixMultiplicationShared
                   19.44%  3.7339ms        42  88.903us  88.672us  89.248us  [CUDA memcpy HtoD]
                   15.77%  3.0282ms        10  302.82us  300.58us  305.28us  matrixMultiplication(int*,
                    8.84%  1.6976ms        21  80.839us  80.800us  80.928us  [CUDA memcpy DtoH]
      API calls:   80.97%  192.62ms         2  96.312ms  3.4600us  192.62ms  cudaEventCreate
                   13.25%  31.528ms        63  500.45us  128.01us  1.8694ms  cudaMemcpy
                    2.35%  5.5946ms        63  88.802us  6.7240us  241.30us  cudaFree
                    2.27%  5.4070ms        63  85.824us  2.2160us  307.09us  cudaMalloc
                    0.75%  1.7858ms       404  4.4200us     413ns  207.82us  cuDeviceGetAttribute
                    0.21%  490.88us        21  23.375us  14.870us  64.508us  cudaLaunchKernel
                    0.10%  238.70us        40  5.9670us  3.2540us  24.190us  cudaEventRecord
                    0.04%  84.646us        20  4.2320us  1.9080us  5.8010us  cudaEventSynchronize
                    0.02%  54.832us         4  13.708us  8.6680us  27.578us  cuDeviceGetName
                    0.02%  43.282us        20  2.1640us  1.3490us  4.3500us  cudaEventElapsedTime
                    0.01%  22.656us         4  5.6640us  3.1760us  12.426us  cuDeviceGetPCIBusId
                    0.00%  4.6180us         8     577ns     393ns  1.2660us  cuDeviceGet
                    0.00%  4.1820us         3  1.3940us     796ns  2.1600us  cuDeviceGetCount
                    0.00%  3.5650us         2  1.7820us     557ns  3.0080us  cudaEventDestroy
                    0.00%  2.6980us         1  2.6980us  2.6980us  2.6980us  cuModuleGetLoadingMode
                    0.00%  2.6290us         4     657ns     537ns     954ns  cuDeviceTotalMem
                    0.00%  2.0920us         4     523ns     480ns     632ns  cuDeviceGetUuid
```

Using shared memory tile size = 8 x 8

```
Using matrix size = 262144 with width = 512
Using shared memory tile size = 8 x 8
==491022== NVPROF is profiling process 491022, command: ./matmul_sharedmem_512x512_8

===== Performance Comparison =====
CPU takes 703.765198ms
GPU Regular takes 0.545574ms
```

```
GPU Shared Memory takes 0.567958ms
Speedup (Shared vs Regular): 0.96x
Speedup (Shared vs CPU): 1239.11x
The matrix mul is right as both CPU and GPU matches
==491022== Profiling application: ./matmul_sharedmem_512x512_8
==491022== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   31.36%  3.7315ms        42  88.845us  88.640us  89.248us  [CUDA memcpy HtoD]
                   28.29%  3.3669ms        11  306.08us  299.97us  307.84us  matrixMultiplicationShared
                   26.08%  3.1041ms        10  310.41us  309.54us  310.88us  matrixMultiplication(int*,
                   14.27%  1.6976ms        21  80.836us  80.767us  81.151us  [CUDA memcpy DtoH]
      API calls:   80.81%  130.82ms         2  65.410ms  1.0520us  130.82ms  cudaEventCreate
                   12.69%  20.543ms        63  326.08us  121.97us  834.79us  cudaMemcpy
                    2.70%  4.3667ms        63  69.312us  5.6680us  142.22us  cudaFree
                    2.27%  3.6703ms        63  58.259us  2.2700us  161.73us  cudaMalloc
                    1.10%  1.7844ms       404  4.4160us     403ns  208.44us  cuDeviceGetAttribute
                    0.20%  325.57us        21  15.503us  12.120us  30.697us  cudaLaunchKernel
                    0.10%  163.46us        40  4.0860us  2.7800us  14.180us  cudaEventRecord
                    0.05%  82.025us        20  4.1010us  1.6580us  4.8070us  cudaEventSynchronize
                    0.04%  70.236us         4  17.559us  8.3630us  30.650us  cuDeviceGetName
                    0.02%  27.783us        20  1.3890us  1.2250us  1.9960us  cudaEventElapsedTime
                    0.01%  21.395us         4  5.3480us  3.0900us  11.267us  cuDeviceGetPCIBusId
                    0.00%  4.3080us         8     538ns     402ns  1.1080us  cuDeviceGet
                    0.00%  4.1070us         3  1.3690us     703ns  2.1570us  cuDeviceGetCount
                    0.00%  3.7440us         2  1.8720us     491ns  3.2530us  cudaEventDestroy
                    0.00%  2.7960us         1  2.7960us  2.7960us  2.7960us  cuModuleGetLoadingMode
                    0.00%  2.7660us         4     691ns     497ns  1.0180us  cuDeviceTotalMem
                    0.00%  2.1590us         4     539ns     480ns     709ns  cuDeviceGetUuid
```

Using shared memory tile size = 16 x 16

```
Using matrix size = 262144 with width = 512
Using shared memory tile size = 16 x 16
==491065== NVPROF is profiling process 491065, command: ./matmul_sharedmem_512x512_16

===== Performance Comparison =====
CPU takes 694.432434ms
GPU Regular takes 0.524624ms
GPU Shared Memory takes 0.448822ms
Speedup (Shared vs Regular): 1.17x
Speedup (Shared vs CPU): 1547.23x
The matrix mul is right as both CPU and GPU matches
==491065== Profiling application: ./matmul_sharedmem_512x512_16
==491065== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   34.69%  3.7393ms        42  89.031us  88.512us  99.072us  [CUDA memcpy HtoD]
                   28.79%  3.1039ms        10  310.39us  309.60us  311.71us  matrixMultiplication(int*,
                   20.77%  2.2391ms        11  203.56us  198.72us  204.61us  matrixMultiplicationShared
                   15.74%  1.6971ms        21  80.812us  80.767us  81.120us  [CUDA memcpy DtoH]
      API calls:   81.23%  125.96ms         2  62.979ms  1.1850us  125.96ms  cudaEventCreate
                   12.13%  18.813ms        63  298.62us  116.58us  705.12us  cudaMemcpy
                    2.75%  4.2603ms        63  67.623us  5.1380us  126.19us  cudaFree
                    2.31%  3.5748ms        63  56.743us  2.1840us  167.71us  cudaMalloc
                    1.16%  1.8004ms       404  4.4560us     425ns  203.61us  cuDeviceGetAttribute
```

```
                    0.21%  318.28us        21  15.156us  11.511us  37.924us  cudaLaunchKernel
                    0.09%  140.81us        40  3.5200us  2.5870us  6.6760us  cudaEventRecord
                    0.05%  82.637us        20  4.1310us  1.7170us  4.5790us  cudaEventSynchronize
                    0.03%  44.251us         4  11.062us  8.5760us  18.305us  cuDeviceGetName
                    0.02%  26.494us        20  1.3240us  1.1620us  2.0130us  cudaEventElapsedTime
                    0.02%  24.280us         4  6.0700us  3.1520us  13.650us  cuDeviceGetPCIBusId
                    0.00%  4.1750us         8     521ns     388ns  1.2000us  cuDeviceGet
                    0.00%  3.5650us         2  1.7820us     445ns  3.1200us  cudaEventDestroy
                    0.00%  3.3420us         3  1.1140us     556ns  2.1000us  cuDeviceGetCount
                    0.00%  2.7440us         4     686ns     512ns     969ns  cuDeviceTotalMem
                    0.00%  2.2630us         4     565ns     480ns     747ns  cuDeviceGetUuid
                    0.00%  1.0920us         1  1.0920us  1.0920us  1.0920us  cuModuleGetLoadingMode
```

Using shared memory tile size = 32 x 32

```
Using matrix size = 262144 with width = 512
Using shared memory tile size = 32 x 32
==491203== NVPROF is profiling process 491203, command: ./matmul_sharedmem_512x512_32

===== Performance Comparison =====
CPU takes 708.010498ms
GPU Regular takes 0.534720ms
GPU Shared Memory takes 0.437165ms
Speedup (Shared vs Regular): 1.22x
Speedup (Shared vs CPU): 1619.55x
The matrix mul is right as both CPU and GPU matches
==491203== Profiling application: ./matmul_sharedmem_512x512_32
==491203== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   35.34%  3.7318ms        42  88.853us  88.640us  89.183us  [CUDA memcpy HtoD]
                   29.42%  3.1070ms        10  310.70us  310.08us  311.39us  matrixMultiplication(int*,
                   19.17%  2.0246ms        11  184.06us  180.54us  185.15us  matrixMultiplicationShared
                   16.07%  1.6972ms        21  80.819us  80.799us  80.960us  [CUDA memcpy DtoH]
      API calls:   81.32%  128.46ms         2  64.229ms  3.0640us  128.45ms  cudaEventCreate
                   12.10%  19.119ms        63  303.48us  127.14us  702.36us  cudaMemcpy
                    2.74%  4.3268ms        63  68.678us  5.7270us  133.21us  cudaFree
                    2.30%  3.6335ms        63  57.674us  2.1660us  170.56us  cudaMalloc
                    1.12%  1.7674ms       404  4.3740us     398ns  202.61us  cuDeviceGetAttribute
                    0.20%  322.25us        21  15.345us  12.607us  31.918us  cudaLaunchKernel
                    0.09%  148.88us        40  3.7210us  2.6760us  7.0830us  cudaEventRecord
                    0.05%  83.452us        20  4.1720us  1.7680us  4.7760us  cudaEventSynchronize
                    0.03%  44.936us         4  11.234us  8.3490us  17.937us  cuDeviceGetName
                    0.02%  27.382us        20  1.3690us  1.1490us  2.3310us  cudaEventElapsedTime
                    0.01%  21.824us         4  5.4560us  2.0050us  13.467us  cuDeviceGetPCIBusId
                    0.00%  4.6940us         2  2.3470us     493ns  4.2010us  cudaEventDestroy
                    0.00%  4.2980us         8     537ns     379ns  1.4930us  cuDeviceGet
                    0.00%  3.8250us         3  1.2750us     685ns  2.4210us  cuDeviceGetCount
                    0.00%  2.5230us         4     630ns     490ns     905ns  cuDeviceTotalMem
                    0.00%  2.0720us         4     518ns     460ns     616ns  cuDeviceGetUuid
                    0.00%     959ns         1     959ns     959ns     959ns  cuModuleGetLoadingMode
```

For a fixed matrix size of 512×512, we vary the shared memory tile size to determine the optimal configuration. The table below summarizes the GPU shared memory execution time for different tile sizes.

Tile Size	GPU Shared Memory Time (ms)
4×4	1.342522
8×8	0.567958
16×16	0.448822
32×32	0.437165

Table 4: GPU Shared Memory Execution Time for Different Tile Sizes

Observations

- As the tile size increases, GPU shared memory execution time decreases.
- The optimal tile size is 32×32, where execution time is the lowest at **0.437165 ms**.
- A larger tile size results in better memory utilization and reduced redundant memory accesses, leading to increased speedup.
- Smaller size maybe slower since we need to sync thread() and loading data from global to shared memory

Based on this analysis, using a tile size of 32×32 provides the best performance for matrix multiplication on shared memory, as the matrix size is big in this case, so bigger tile works well with it. However, larger tiles may get each block requires more shared memory and potentially more registers, which reduces the number of thread blocks that can run concurrently on a single multiprocessor.