

## Memory Assignment

Module 5

09/28/2025

Annabel Lin

### About the submission

The program I created generates a random array of size = totalThreads and runs a filter kernel over it. I made a version that uses shared memory, and a version that does not use device shared memory.

Host memory usage: malloc float arrays for input/results, used when data is transferred to/from host and gpu

Global memory usage: cudaMalloc float arrays for input/results, used by the gpu when reading input and writing results. Input is copied from host to gpu, and gpu results are copied from gpu to host.

Shared memory usage: The shared memory version of the kernel populates a tile for values that may be reused. Values are copied from global memory to shared memory, and then the shared memory is used for accessing the data.

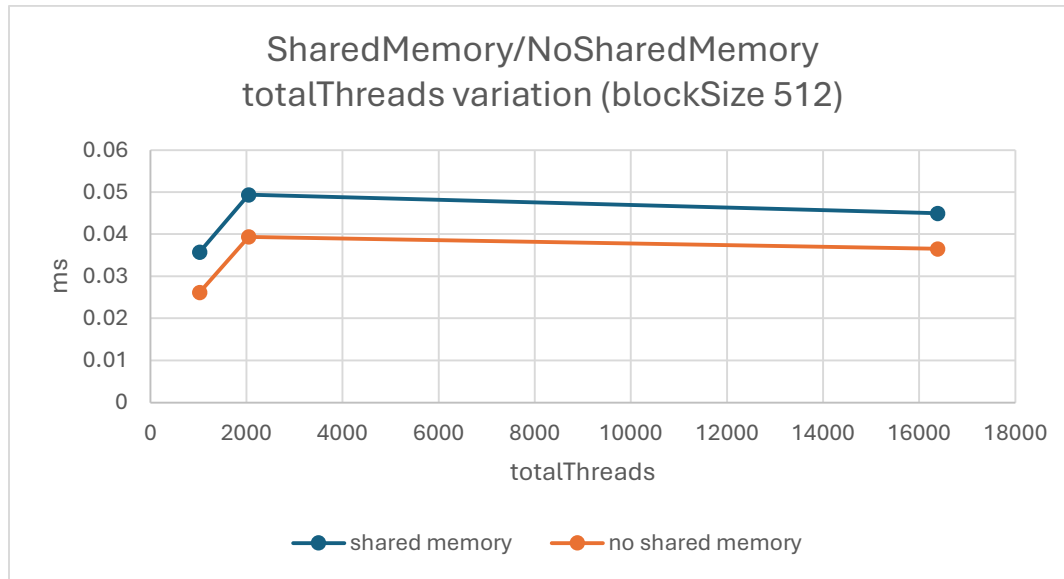
Constant memory usage: The filter array is put in constant memory since it is the same for every thread.

Register memory usage: The local variables used in the \_\_global\_\_ function are stored in register memory. During compilation, I see that local registers are used (varying depending on my filter kernel size)

```
C:\Users\Annabel\Documents\JHU EP Masters\GPU Programming\alin84-EN605.617\module5\assignment_submission>make
nvcc -Xptxas -v assignment.cu -o assignment.exe
assignment.cu
ptxas info      : 0 bytes gmem, 44 bytes cmem[3]
ptxas info      : Compiling entry function '_Z13warmup_kernelPfi' for 'sm_75'
ptxas info      : Function properties for _Z13warmup_kernelPfi
0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info      : Used 8 registers, used 0 barriers, 364 bytes cmem[0]
ptxas info      : Compile time = 0.000 ms
ptxas info      : Compiling entry function '_Z26filter_kernel_no_sharedmemPKfPfi' for 'sm_75'
ptxas info      : Function properties for _Z26filter_kernel_no_sharedmemPKfPfi
0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info      : Used 18 registers, used 0 barriers, 372 bytes cmem[0]
ptxas info      : Compile time = 0.000 ms
ptxas info      : Compiling entry function '_Z25filter_kernel_w_sharedmemPKfPfi' for 'sm_75'
ptxas info      : Function properties for _Z25filter_kernel_w_sharedmemPKfPfi
0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info      : Used 24 registers, used 1 barriers, 372 bytes cmem[0]
ptxas info      : Compile time = 0.000 ms
tmpxft_00002c74_00000000-7_assignment.cudaf1.cpp
```

The results are for a filter kernel of radius 5 (size 11).

## Varying threads

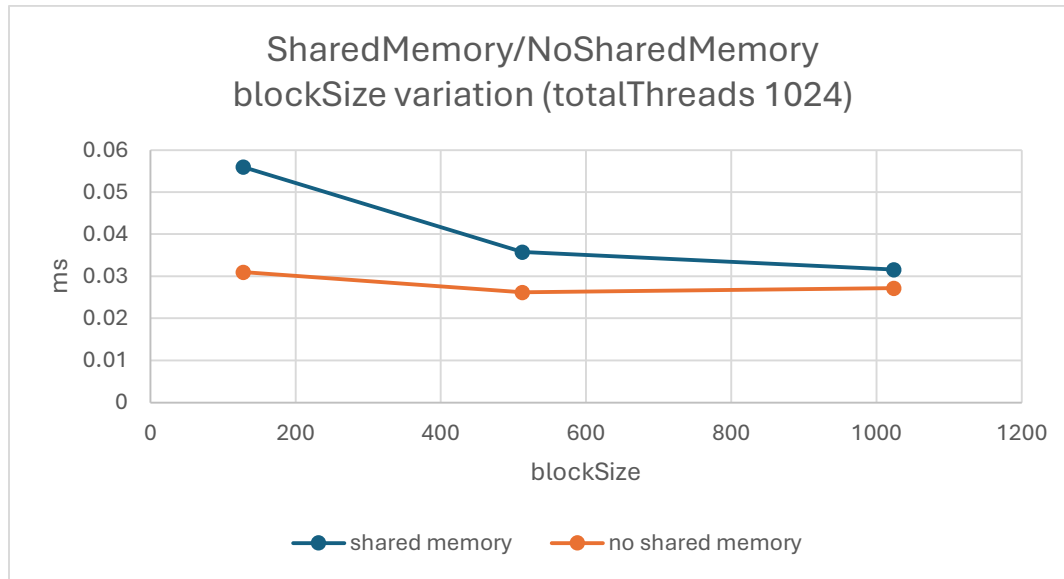


General trend: Increasing totalThreads from 1024 to 2048 had a large increase in execution time, which could be due to a larger workload for the GPU and any initial overhead.

Increasing totalThreads from 2048 to 16,384 had a small decrease in execution time, but the graph is mostly flat – which could mean that this is just about as parallel/optimized that the GPU can get for our operations.

Shared vs no shared memory: For totalThreads variation, the version using shared memory was consistently slower than the one that does not utilize shared memory. The most likely reason is that the setup for shared memory (loading the values in from global) cost more overhead than the benefit of having quicker access.

## Varying block size



General trend: Increasing blockSize from 128 to 256 significantly increased performance. This could be because this increased occupancy led to more efficient usage of the gpu. Increasing blockSize from 256 to 1024 had less of an effect on performance, general plateau. This could be because the gpu is already saturated and increasing blockSize doesn't affect it much.

Shared vs no shared memory: The graph shapes are slightly different for shared vs no shared memory.

The execution time for shared memory is much higher at blockSize 128 because there is the most overhead and perhaps the least reuse of data. The shared memory execution time improves between blockSize 512 and 1024 perhaps due to effective reuse of the data it loaded, but it still does not beat the non-shared memory version.

The execution time for no shared memory remains relatively consistent as blockSize varies. This may be because accessing global memory does not really depend on the blockSize, and that is probably the biggest cost.

## Shared Memory vs No Shared Memory

Generally, utilizing shared memory performed worse than not using shared memory. This is likely because using shared memory adds overhead in loading elements – it needs to check for boundaries and populate the elements with the correct values, and then synchronize the threads before computation. It seems that if we wanted to utilize shared memory to

improve efficiency, the program would have to have more repeated access to the shared memory. The current program's cost to benefit ratio of using shared memory is too large.



```

C:\Users\Annabel\Documents\JHU EP Masters\GPU Programming\alin84-EN605.617\module5\assignment_submission>assignment.exe 1024 1024
Run w totalThreads 1024, blockSize 1024, numBlocks 1, filterSize 11

Kernel execution times: shared=0.032 ms, no shared=0.025 ms

C:\Users\Annabel\Documents\JHU EP Masters\GPU Programming\alin84-EN605.617\module5\assignment_submission>assignment.exe 1024 1024
Run w totalThreads 1024, blockSize 1024, numBlocks 1, filterSize 11

Kernel execution times: shared=0.017 ms, no shared=0.029 ms

C:\Users\Annabel\Documents\JHU EP Masters\GPU Programming\alin84-EN605.617\module5\assignment_submission>assignment.exe 1024 1024
Run w totalThreads 1024, blockSize 1024, numBlocks 1, filterSize 11

Kernel execution times: shared=0.024 ms, no shared=0.033 ms

C:\Users\Annabel\Documents\JHU EP Masters\GPU Programming\alin84-EN605.617\module5\assignment_submission>assignment.exe 1024 1024
Run w totalThreads 1024, blockSize 1024, numBlocks 1, filterSize 11

Kernel execution times: shared=0.034 ms, no shared=0.022 ms

C:\Users\Annabel\Documents\JHU EP Masters\GPU Programming\alin84-EN605.617\module5\assignment_submission>assignment.exe 1024 1024
Run w totalThreads 1024, blockSize 1024, numBlocks 1, filterSize 11

Kernel execution times: shared=0.051 ms, no shared=0.027 ms

C:\Users\Annabel\Documents\JHU EP Masters\GPU Programming\alin84-EN605.617\module5\assignment_submission>
C:\Users\Annabel\Documents\JHU EP Masters\GPU Programming\alin84-EN605.617\module5\assignment_submission>assignment.exe 1024 128
Run w totalThreads 1024, blockSize 128, numBlocks 8, filterSize 11

Kernel execution times: shared=0.085 ms, no shared=0.051 ms

C:\Users\Annabel\Documents\JHU EP Masters\GPU Programming\alin84-EN605.617\module5\assignment_submission>assignment.exe 1024 128
Run w totalThreads 1024, blockSize 128, numBlocks 8, filterSize 11

Kernel execution times: shared=0.080 ms, no shared=0.027 ms

C:\Users\Annabel\Documents\JHU EP Masters\GPU Programming\alin84-EN605.617\module5\assignment_submission>assignment.exe 1024 128
Run w totalThreads 1024, blockSize 128, numBlocks 8, filterSize 11

Kernel execution times: shared=0.034 ms, no shared=0.022 ms

C:\Users\Annabel\Documents\JHU EP Masters\GPU Programming\alin84-EN605.617\module5\assignment_submission>assignment.exe 1024 128
Run w totalThreads 1024, blockSize 128, numBlocks 8, filterSize 11

Kernel execution times: shared=0.048 ms, no shared=0.033 ms

C:\Users\Annabel\Documents\JHU EP Masters\GPU Programming\alin84-EN605.617\module5\assignment_submission>assignment.exe 1024 128
Run w totalThreads 1024, blockSize 128, numBlocks 8, filterSize 11

Kernel execution times: shared=0.033 ms, no shared=0.022 ms

```

## Chart generation

