Analysis Report

maxwell_scudnn_128x32_stridedB_small_nn

Duration	954.65 µs
Grid Size	[784,1,1]
Block Size	[128,1,1]
Registers/Thread	92
Shared Memory/Block	10 KiB
Shared Memory Requested	96 KiB
Shared Memory Executed	96 KiB
Shared Memory Bank Size	4 B

[0] GeForce GTX 1080

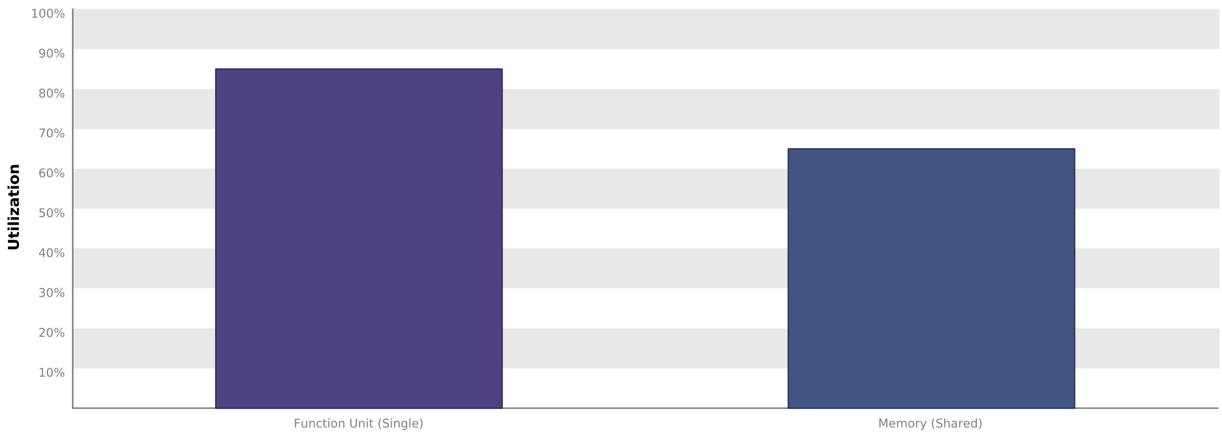
GPU UUID	GPU-edd19385-a5f1-ce46-e1d9-61408827cccc
Compute Capability	6.1
Max. Threads per Block	1024
Max. Threads per Multiprocessor	2048
Max. Shared Memory per Block	48 KiB
Max. Shared Memory per Multiprocessor	96 KiB
Max. Registers per Block	65536
Max. Registers per Multiprocessor	65536
Max. Grid Dimensions	[2147483647, 65535, 65535]
Max. Block Dimensions	[1024, 1024, 64]
Max. Warps per Multiprocessor	64
Max. Blocks per Multiprocessor	32
Half Precision FLOP/s	72.9 GigaFLOP/s
Single Precision FLOP/s	9.331 TeraFLOP/s
Double Precision FLOP/s	291.6 GigaFLOP/s
Number of Multiprocessors	20
Multiprocessor Clock Rate	1.823 GHz
Concurrent Kernel	true
Max IPC	6
Threads per Warp	32
Global Memory Bandwidth	320.32 GB/s
Global Memory Size	7.923 GiB
Constant Memory Size	64 KiB
L2 Cache Size	2 MiB
Memcpy Engines	2
PCIe Generation	3
PCIe Link Rate	8 Gbit/s
PCIe Link Width	16

1. Compute, Bandwidth, or Latency Bound

The first step in analyzing an individual kernel is to determine if the performance of the kernel is bounded by computation, memory bandwidth, or instruction/memory latency. The results below indicate that the performance of kernel "maxwell_scudnn_128x32_strid..." is most likely limited by compute. You should first examine the information in the "Compute Resources" section to determine how it is limiting performance.

1.1. Kernel Performance Is Bound By Compute

For device "GeForce GTX 1080" the kernel's memory utilization is significantly lower than its compute utilization. These utilization levels indicate that the performance of the kernel is most likely being limited by computation on the SMs.



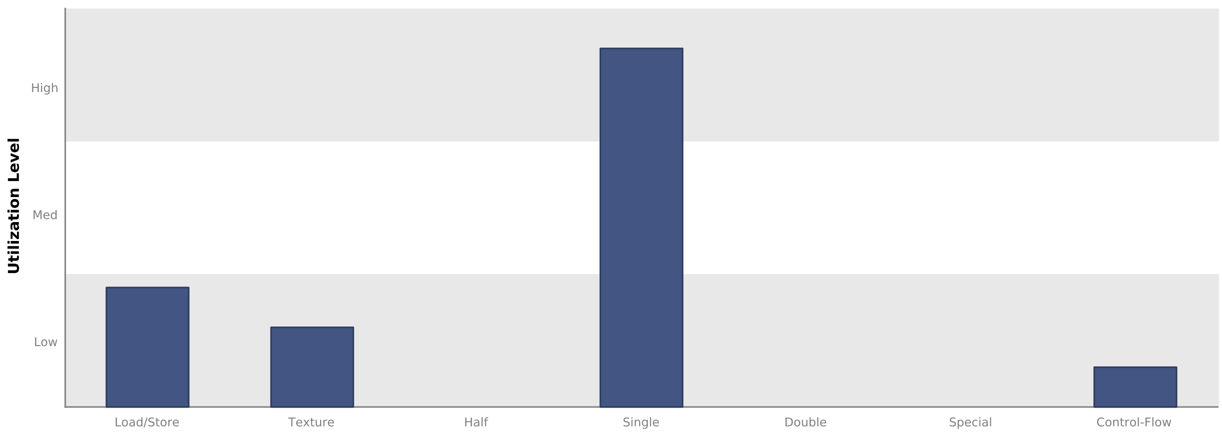
2. Compute Resources

GPU compute resources limit the performance of a kernel when those resources are insufficient or poorly utilized. Compute resources are used most efficiently when instructions do not overuse a function unit. The results below indicate that compute performance may be limited by overuse of a function unit.

2.1. GPU Utilization Is Limited By Function Unit Usage

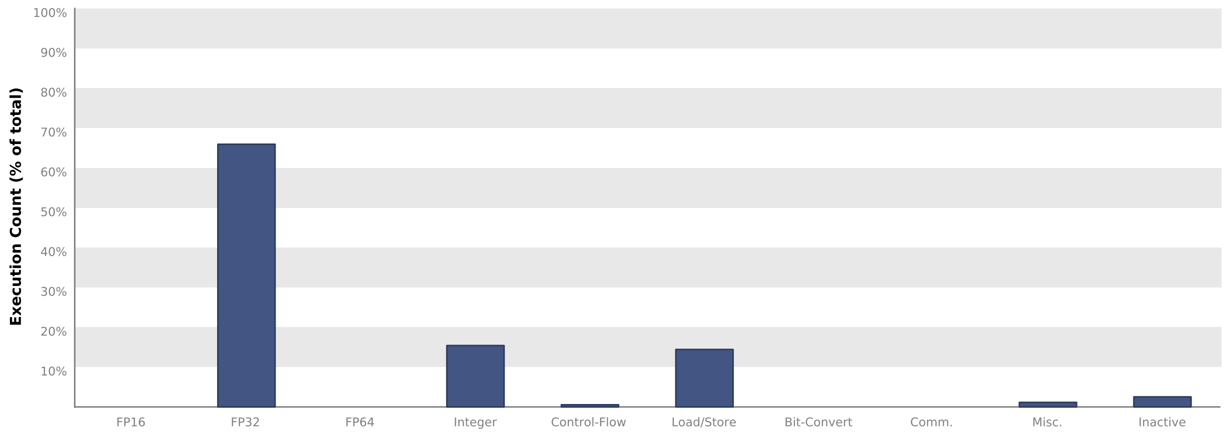
Different types of instructions are executed on different function units within each SM. Performance can be limited if a function unit is over-used by the instructions executed by the kernel. The following results show that the kernel's performance is potentially limited by overuse of the following function units: Single.

Load/Store - Load and store instructions for shared and constant memory.
Texture - Load and store instructions for local, global, and texture memory.
Half - Half-precision floating-point arithmetic instructions.
Single - Single-precision integer and floating-point arithmetic instructions.
Double - Double-precision floating-point arithmetic instructions.
Special - Special arithmetic instructions such as sin, cos, popc, etc.
Control-Flow - Direct and indirect branches, jumps, and calls.



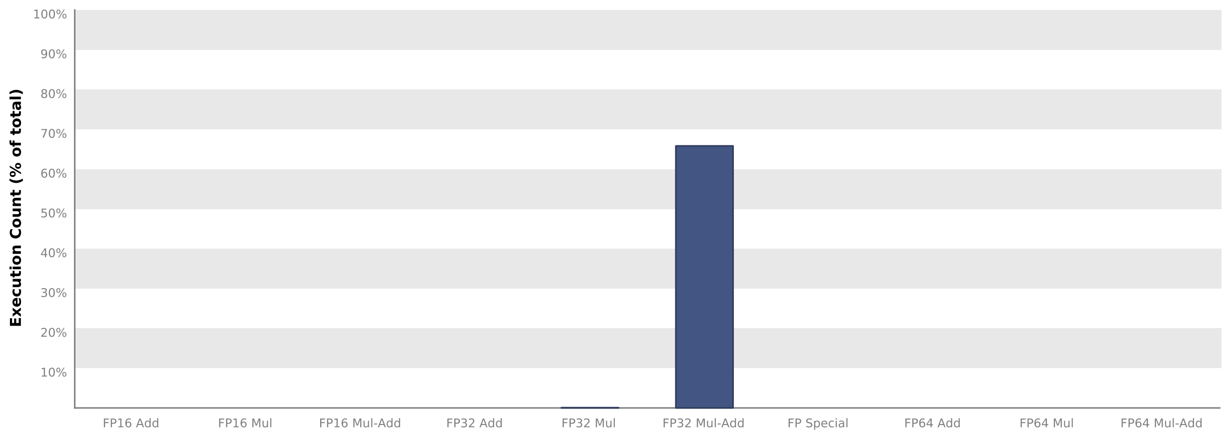
2.2. Instruction Execution Counts

The following chart shows the mix of instructions executed by the kernel. The instructions are grouped into classes and for each class the chart shows the percentage of thread execution cycles that were devoted to executing instructions in that class. The "Inactive" result shows the thread executions that did not execute any instruction because the thread was predicated or inactive due to divergence.



2.3. Floating-Point Operation Counts

The following chart shows the mix of floating-point operations executed by the kernel. The operations are grouped into classes and for each class the chart shows the percentage of thread execution cycles that were devoted to executing operations in that class. The results do not sum to 100% because non-floating-point operations executed by the kernel are not shown in this chart.



3. Memory Bandwidth

Memory bandwidth limits the performance of a kernel when one or more memories in the GPU cannot provide data at the rate requested by the kernel. The results below indicate that the kernel is limited by the bandwidth available to the shared memory.

3.1. Shared Memory Alignment and Access Pattern

Memory bandwidth is used most efficiently when each shared memory load and store has proper alignment and access pattern.

Optimization: Select each entry below to open the source code to a shared load or store within the kernel with an inefficient alignment or access pattern. For each load or store improve the alignment and access pattern of the memory access.

3.2. Memory Bandwidth And Utilization

The following table shows the memory bandwidth used by this kernel for the various types of memory on the device. The table also shows the utilization of each memory type relative to the maximum throughput supported by the memory.

Transactions	Bandwidth	Utilization	
Shared Memory			
Shared Loads	15830528	2,405.693 GB/s	
Shared Stores	3352384	509.446 GB/s	
Shared Total	19182912	2,915.139 GB/s	Idle Low Medium High Max
L2 Cache			
Reads	1642617	62.405 GB/s	
Writes	12557	477.057 MB/s	
Total	1655174	62.882 GB/s	Idle Low Medium High Max
Unified Cache			
Local Loads	0	0 B/s	
Local Stores	0	0 B/s	
Global Loads	22973824	870.066 GB/s	
Global Stores	12544	476.563 MB/s	
Texture Reads	9986432	379.398 GB/s	
Unified Total	32972800	1,249.941 GB/s	Idle Low Medium High Max
Device Memory			
Reads	401810	15.265 GB/s	
Writes	40301	1.531 GB/s	
Total	442111	16.796 GB/s	Idle Low Medium High Max
System Memory			
[PCIe configuration: Gen3 x16, 8 Gbit/s]			
Reads	0	0 B/s	Idle Low Medium High Max
Writes	5	189.956 kB/s	Idle Low Medium High Max

4. Instruction and Memory Latency

Instruction and memory latency limit the performance of a kernel when the GPU does not have enough work to keep busy. The performance of latency-limited kernels can often be improved by increasing occupancy. Occupancy is a measure of how many warps the kernel has active on the GPU, relative to the maximum number of warps supported by the GPU. Theoretical occupancy provides an upper bound while achieved occupancy indicates the kernel's actual occupancy. The results below indicate that occupancy can be improved by reducing the number of registers used by the kernel.

4.1. GPU Utilization Is Limited By Register Usage

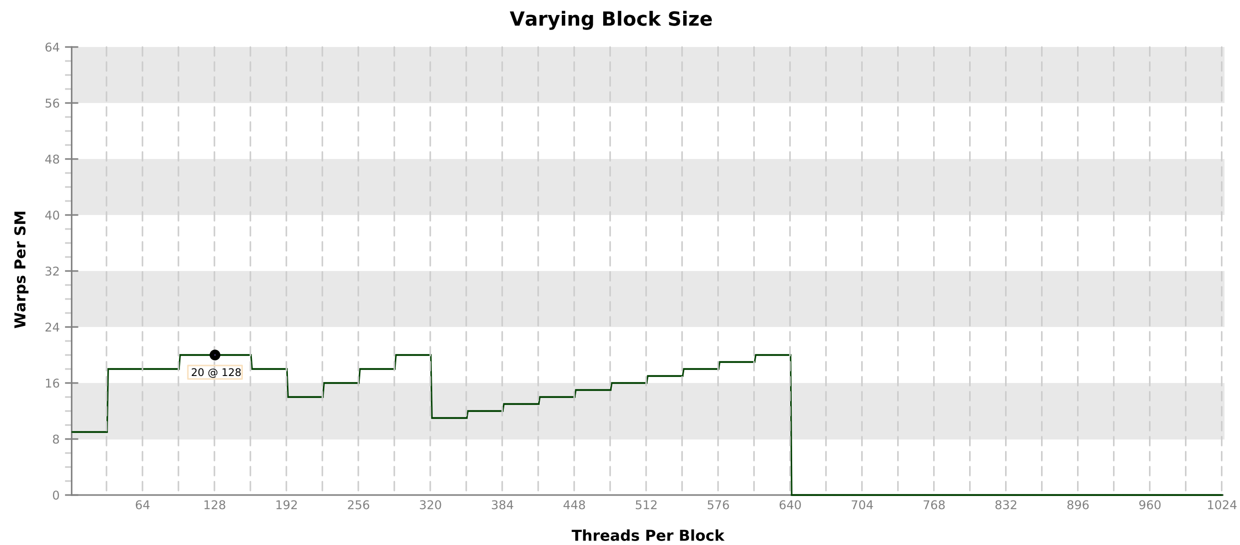
The kernel uses 92 registers for each thread (11776 registers for each block). This register usage is likely preventing the kernel from fully utilizing the GPU. Device "GeForce GTX 1080" provides up to 65536 registers for each block. Because the kernel uses 11776 registers for each block each SM is limited to simultaneously executing 5 blocks (20 warps). Chart "Varying Register Count" below shows how changing register usage will change the number of blocks that can execute on each SM.

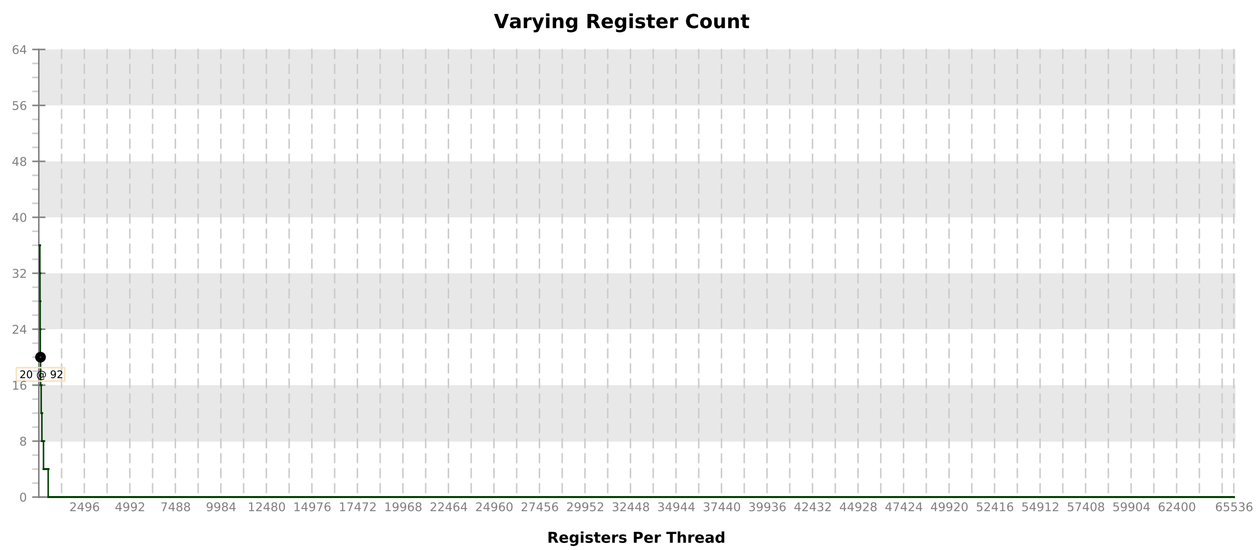
Optimization: Use the -maxrregcount flag or the __launch_bounds__ qualifier to decrease the number of registers used by each thread. This will increase the number of blocks that can execute on each SM. On devices with Compute Capability 5.2 turning global cache off can increase the occupancy limited by register usage.

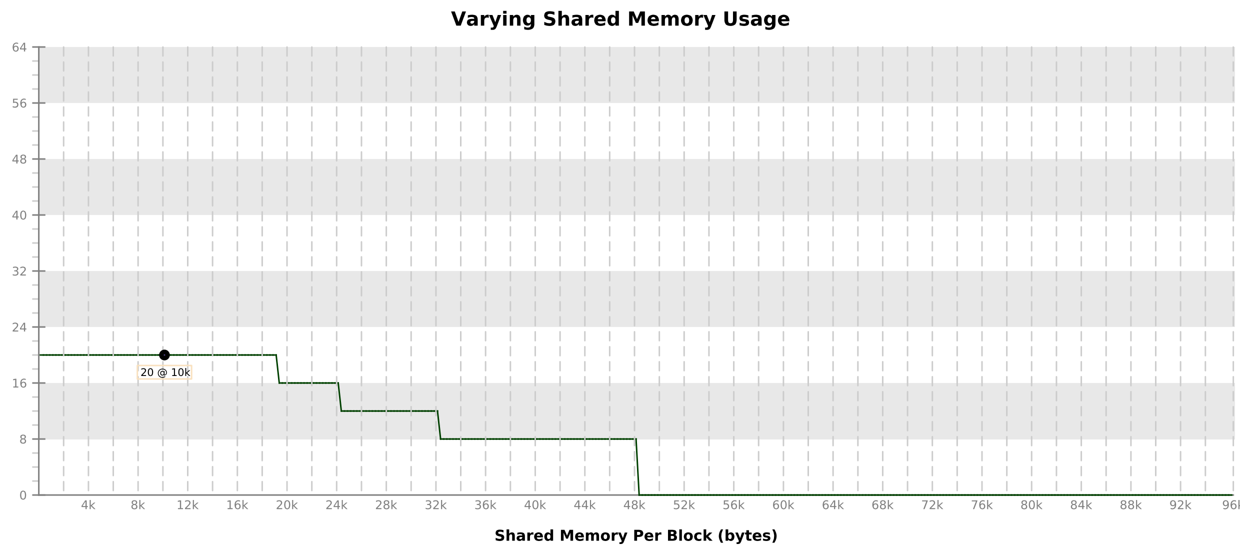
Variable	Achieved	Theoretical	Device Limit	Grid Size: [784,1,1] (784 blocks) Block Size: [128,1,1]
Occupancy Per SM				
Active Blocks		5	32	0 4 8 12 16 20 24 28 32
Active Warps	19.59	20	64	0 9 18 27 36 45 54 63 64
Active Threads		640	2048	0 512 1024 1536 2048
Occupancy	30.6%	31.2%	100%	0% 25% 50% 75% 100%
Warps				
Threads/Block		128	1024	0 256 512 768 1024
Warps/Block		4	32	0 4 8 12 16 20 24 28 32
Block Limit		16	32	0 4 8 12 16 20 24 28 32
Registers				
Registers/Thread		92	65536	0 16384 32768 49152 65536
Registers/Block		12288	65536	0 16k 32k 48k 64k
Block Limit		5	32	0 4 8 12 16 20 24 28 32
Shared Memory				
Shared Memory/Block		10240	98304	0 32k 64k 96k
Block Limit		9	32	0 4 8 12 16 20 24 28 32

4.2. Occupancy Charts

The following charts show how varying different components of the kernel will impact theoretical occupancy.


Varying Block Size
Warps Per SM
Threads Per Block
20 @ 128
0
8
16
24
32
40
48
56
64
64
128
192
256
320
384
448
512
576
640
704
768
832
896
960
1024


Varying Register Count
Registers Per Thread
20 @ 92
0
8
16
24
32
40
48
56
64
2496
4992
7488
9984
12480
14976
17472
19968
22464
24960
27456
29952
32448
34944
37440
39936
42432
44928
47424
49920
52416
54912
57408
59904
62400
65536



Varying Shared Memory Usage
64
56
48
40
32
24
16
8
0
20 @ 10k
4k
8k
12k
16k
20k
24k
28k
32k
36k
40k
44k
48k
52k
56k
60k
64k
68k
72k
76k
80k
84k
88k
92k
96k
Shared Memory Per Block (bytes)