Analysis Report

compute_util_table_ver_1(int*, int**, unsigned int, unsigned int, int, int, int, int, int)

Duration	20.751 ms (20,750,542 ns)
Grid Size	[78125,1,1]
Block Size	[128,1,1]
Registers/Thread	30
Shared Memory/Block	2.543 KiB
Shared Memory Requested	48 KiB
Shared Memory Executed	48 KiB
Shared Memory Bank Size	4 B

[0] GeForce GTX TITAN

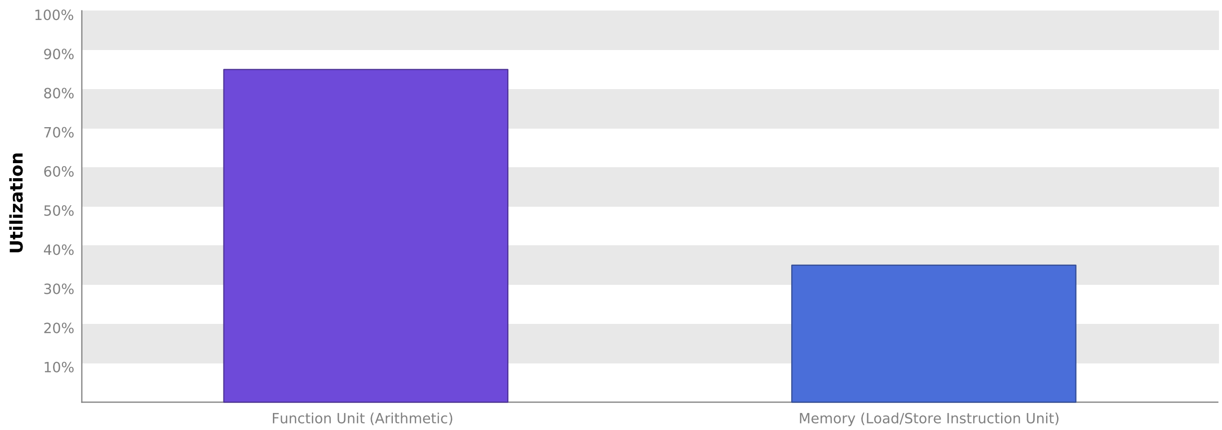
GPU UUID	GPU-c2e0ee17-52b9-35a2-645d-ea8f92144945
Compute Capability	3.5
Max. Threads per Block	1024
Max. Shared Memory per Block	48 KiB
Max. Registers per Block	65536
Max. Grid Dimensions	[2147483647, 65535, 65535]
Max. Block Dimensions	[1024, 1024, 64]
Max. Warps per Multiprocessor	64
Max. Blocks per Multiprocessor	16
Single Precision FLOP/s	4.707 TeraFLOP/s
Double Precision FLOP/s	196.112 GigaFLOP/s
Number of Multiprocessors	14
Multiprocessor Clock Rate	875.5 MHz
Concurrent Kernel	true
Max IPC	7
Threads per Warp	32
Global Memory Bandwidth	288.384 GB/s
Global Memory Size	5.999 GiB
Constant Memory Size	64 KiB
L2 Cache Size	1.5 MiB
Memcpy Engines	1
PCIe Generation	2
PCIe Link Rate	5 Gbit/s
PCIe Link Width	8

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 "compute_util_table_ver_1" 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 TITAN" 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.



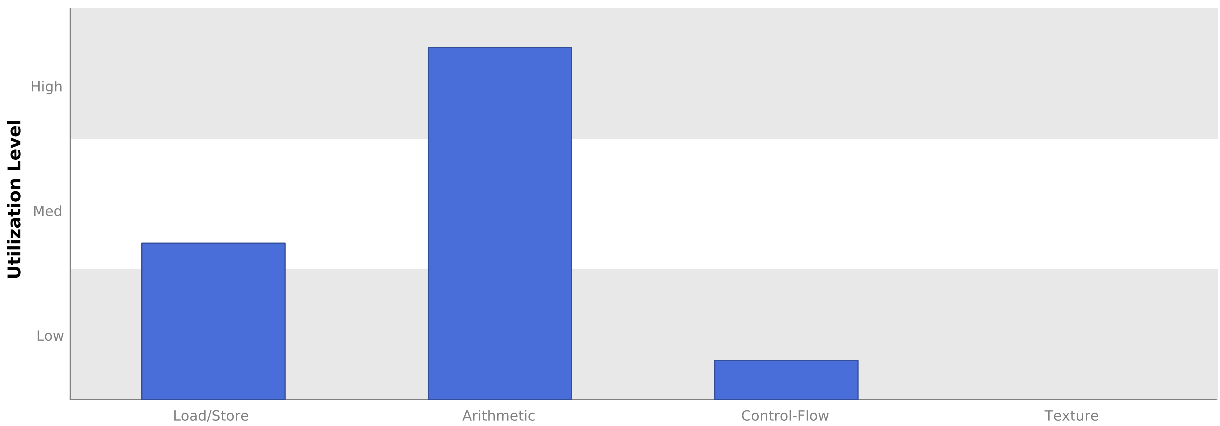
2. Compute Resources

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

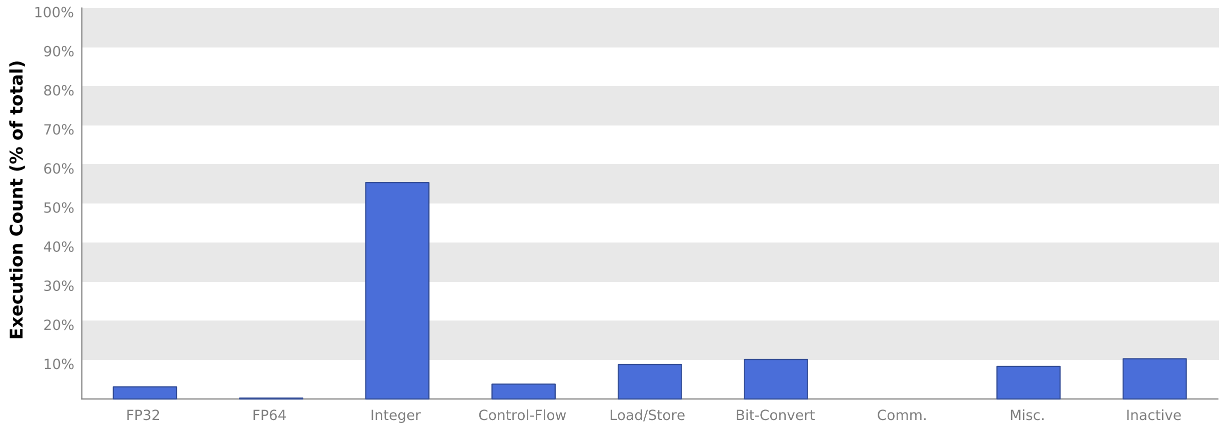
Different types of instructions are executed on different function units within each SM. Performance can be limited if a function unit is over-used by the instructions executed by the kernel. The following results show that the kernel's performance is potentially limited by overuse of the following function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

Load/Store - Load and store instructions for local, shared, global, constant, etc. memory.
Arithmetic - All arithmetic instructions including integer and floating-point add and multiply, logical and binary operations, etc.
Control-Flow - Direct and indirect branches, jumps, and calls.
Texture - Texture operations.



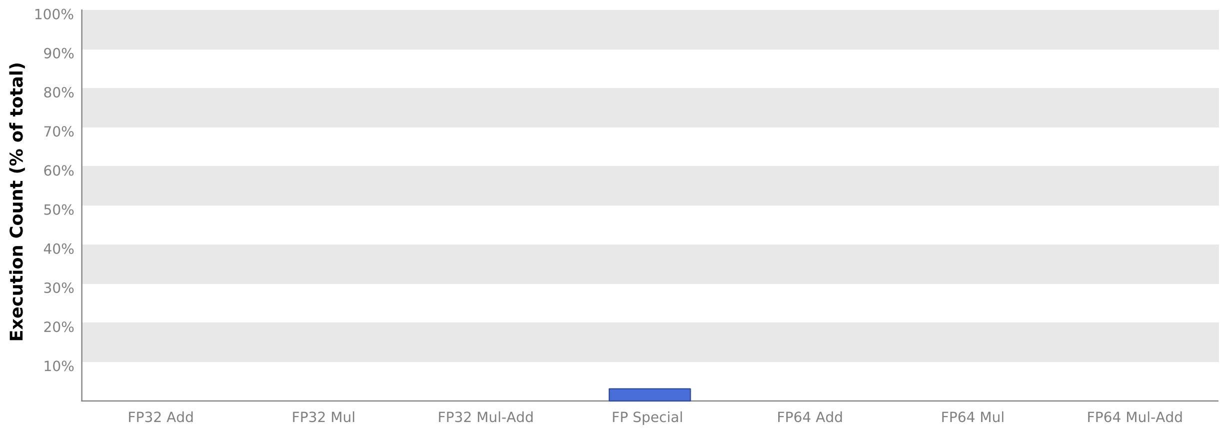
2.2. Instruction Execution Counts

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.

3.1. 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	
L1/Shared Memory			
Local Loads	0	0 B/s	
Local Stores	0	0 B/s	
Shared Loads	24687500	365.687 GB/s	
Shared Stores	14375000	212.932 GB/s	
Global Loads	22179578	50.286 GB/s	
Global Stores	312500	2.314 GB/s	
Atomic	0	0 B/s	
L1/Shared Total	61554578	631.219 GB/s	Idle Low Medium High Max
L2 Cache			
L1 Reads	27158237	50.286 GB/s	
L1 Writes	1250000	2.314 GB/s	
Texture Reads	0	0 B/s	
Noncoherent Reads	0	0 B/s	
Atomic	0	0 B/s	
Total	28408237	52.6 GB/s	Idle Low Medium High Max
Texture Cache			
Reads	0	0 B/s	Idle Low Medium High Max
Device Memory			
Reads	1291231	2.391 GB/s	
Writes	1250001	2.314 GB/s	
Total	2541232	4.705 GB/s	Idle Low Medium High Max
System Memory			
[PCIe configuration: Gen2 x8, 5 Gbit/s]			
Reads	0	0 B/s	Idle Low Medium High Max
Writes	1	1.851 kB/s	Idle Low Medium High Max

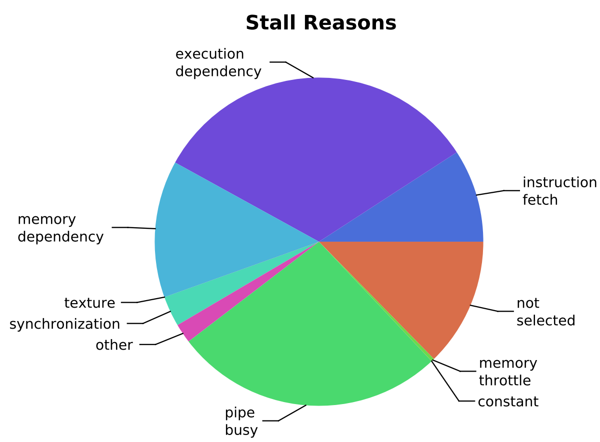
4. Instruction and Memory Latency

Instruction and memory latency limit the performance of a kernel when the GPU does not have enough work to keep busy. The results below indicate that the GPU does not have enough work because instruction execution is stalling excessively.

4.1. Instruction Latencies May Be Limiting Performance

Instruction stall reasons indicate the condition that prevents warps from executing on any given cycle. The following chart shows the break-down of stalls reasons averaged over the entire execution of the kernel. The kernel has good theoretical and achieved occupancy indicating that there are likely sufficient warps executing on each SM. Since occupancy is not an issue it is likely that performance is limited by the instruction stall reasons described below.
Not Selected - Warp was ready to issue, but some other warp issued instead. You may be able to sacrifice occupancy without impacting latency hiding and doing so may help improve cache hit rates.
Texture - The texture sub-system is fully utilized or has too many outstanding requests.
Instruction Fetch - The next assembly instruction has not yet been fetched.
Execution Dependency - An input required by the instruction is not yet available. Execution dependency stalls can potentially be reduced by increasing instruction-level parallelism.
Synchronization - The warp is blocked at a __syncthreads() call.
Pipeline Busy - The compute resource(s) required by the instruction is not yet available.
Memory Dependency - A load/store cannot be made because the required resources are not available or are fully utilized, or too many requests of a given type are outstanding. Data request stalls can potentially be reduced by optimizing memory alignment and access patterns.
Constant - A constant load is blocked due to a miss in the constants cache.
Memory Throttle - Large number of pending memory operations prevent further forward progress. These can be reduced by combining several memory transactions into one.

Optimization: Resolve the primary stall issue; execution dependency.



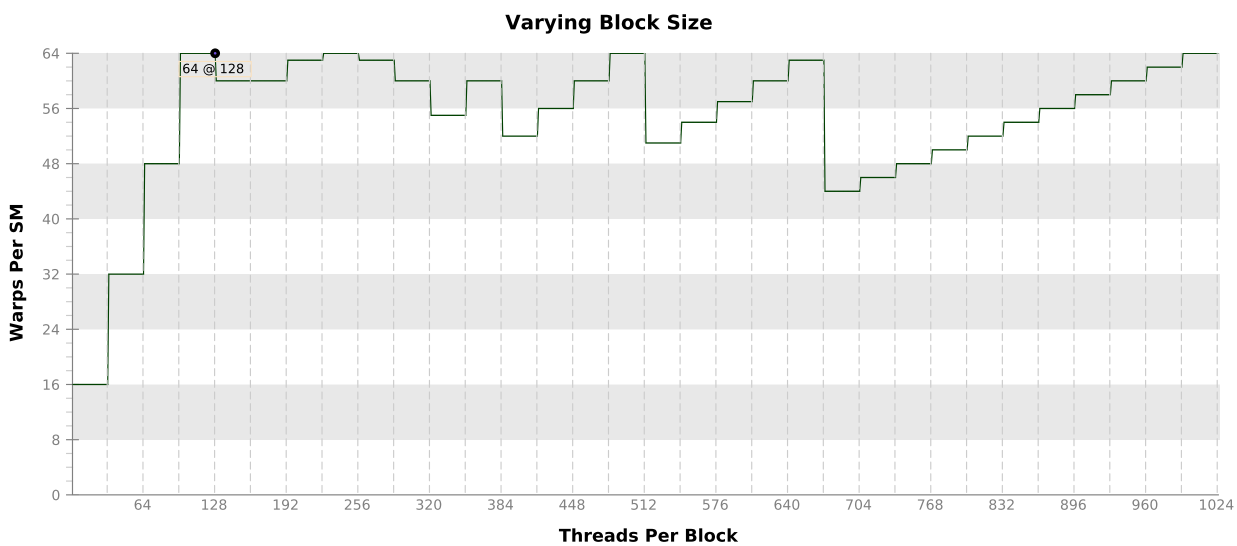
4.2. Occupancy Is Not Limiting Kernel Performance

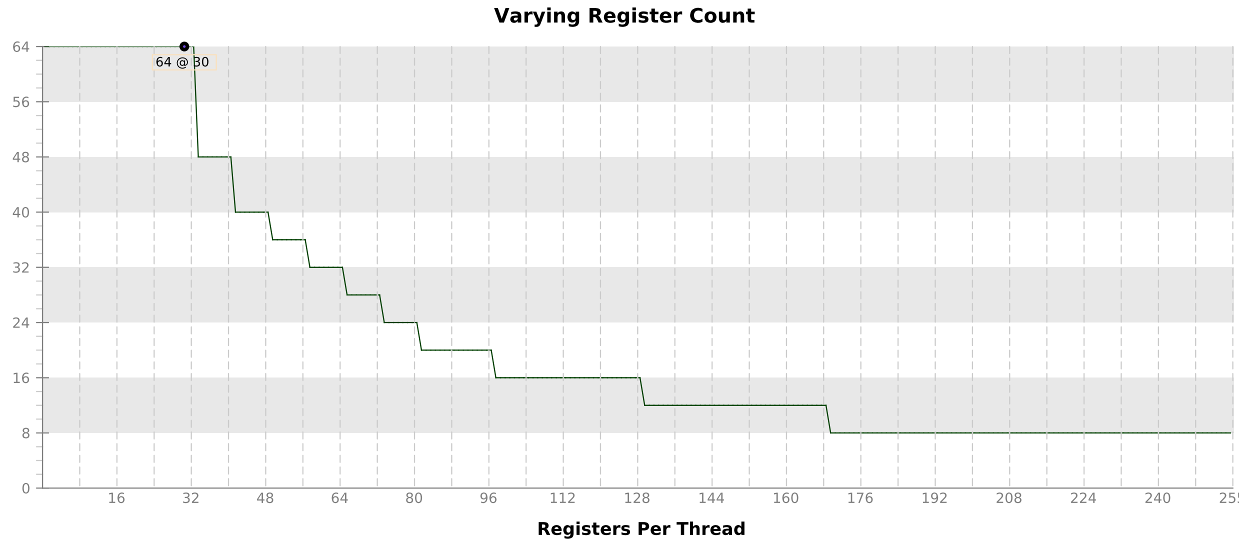
The kernel's block size, register usage, and shared memory usage allow it to fully utilize all warps on the GPU.

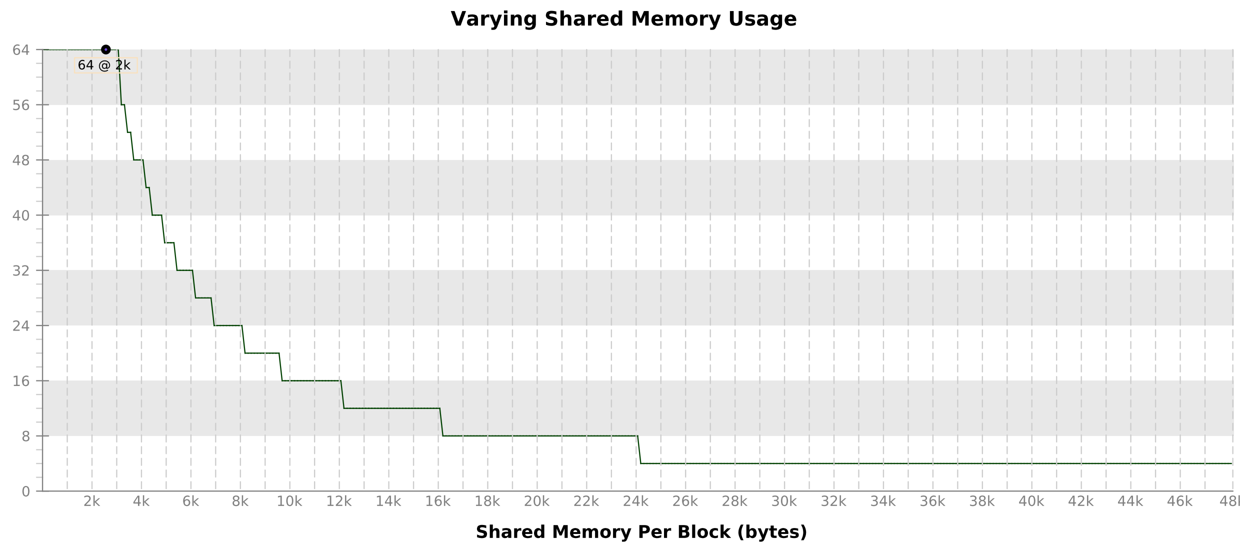
Variable	Achieved	Theoretical	Device Limit	Grid Size: [78125,1,1] (78125 blocks) Block Size: [128
Occupancy Per SM				
Active Blocks		16	16	0 2 4 6 8 10 12 14 16
Active Warps	63.06	64	64	0 9 18 27 36 45 54 63 64
Active Threads		2048	2048	0 512 1024 1536 2048
Occupancy	98.5%	100%	100%	0% 25% 50% 75% 100%
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	16	0 2 4 6 8 10 12 14 16
Registers				
Registers/Thread		30	255	0 64 128 192 255
Registers/Block		4096	65536	0 16k 32k 48k 64k
Block Limit		16	16	0 2 4 6 8 10 12 14 16
Shared Memory				
Shared Memory/Block		2604	49152	0 16k 32k 48k
Block Limit		17	16	0 2 4 6 8 10 12 14 16

4.3. Occupancy Charts

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




Varying Register Count
64 @ 30
0
8
16
24
32
40
48
56
64
16
32
48
64
80
96
112
128
144
160
176
192
208
224
240
255
Registers Per Thread


Varying Shared Memory Usage
64 @ 2k
0
8
16
24
32
40
48
56
64
2k
4k
6k
8k
10k
12k
14k
16k
18k
20k
22k
24k
26k
28k
30k
32k
34k
36k
38k
40k
42k
44k
46k
48k
Shared Memory Per Block (bytes)