Analysis Report

kernelReduceGroupShfl(float*, float*, int, int)

Duration	35.778 µs
Grid Size	[4,128,1]
Block Size	[32,32,1]
Registers/Thread	9
Shared Memory/Block	0 B
Shared Memory Executed	96 KiB
Shared Memory Bank Size	4 B

[0] GeForce GTX 1060 6GB

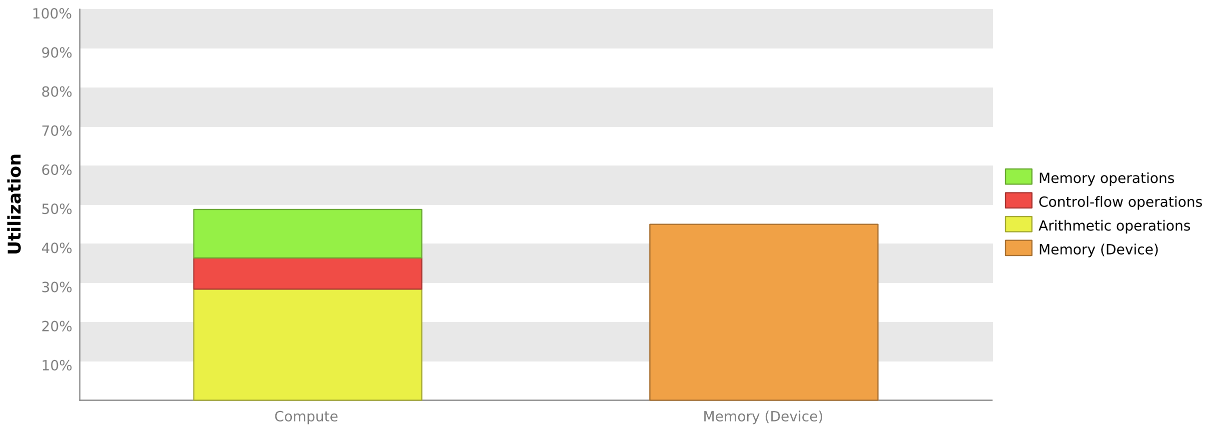
GPU UUID	GPU-eb2e304c-810c-cf47-9ddc-b9f1ccdb795a
Compute Capability	6.1
Max. Threads per Block	1024
Max. Threads per Multiprocessor	2048
Max. Shared Memory per Block	48 KiB
Max. Shared Memory per Multiprocessor	96 KiB
Max. Registers per Block	65536
Max. Registers per Multiprocessor	65536
Max. Grid Dimensions	[2147483647, 65535, 65535]
Max. Block Dimensions	[1024, 1024, 64]
Max. Warps per Multiprocessor	64
Max. Blocks per Multiprocessor	32
Half Precision FLOP/s	34.17 GigaFLOP/s
Single Precision FLOP/s	4.374 TeraFLOP/s
Double Precision FLOP/s	136.68 GigaFLOP/s
Number of Multiprocessors	10
Multiprocessor Clock Rate	1.708 GHz
Concurrent Kernel	true
Max IPC	6
Threads per Warp	32
Global Memory Bandwidth	192.192 GB/s
Global Memory Size	5.935 GiB
Constant Memory Size	64 KiB
L2 Cache Size	1.5 MiB
Memcpy Engines	2
PCIe Generation	2
PCIe Link Rate	5 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 "kernelReduceGroupShfl" is most likely limited by instruction and memory latency. You should first examine the information in the "Instruction And Memory Latency" section to determine how it is limiting performance.

1.1. Kernel Performance Is Bound By Instruction And Memory Latency

This kernel exhibits low compute throughput and memory bandwidth utilization relative to the peak performance of "GeForce GTX 1060 6GB". These utilization levels indicate that the performance of the kernel is most likely limited by the latency of arithmetic or memory operations. Achieved compute throughput and/or memory bandwidth below 60% of peak typically indicates latency issues.



2. 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.

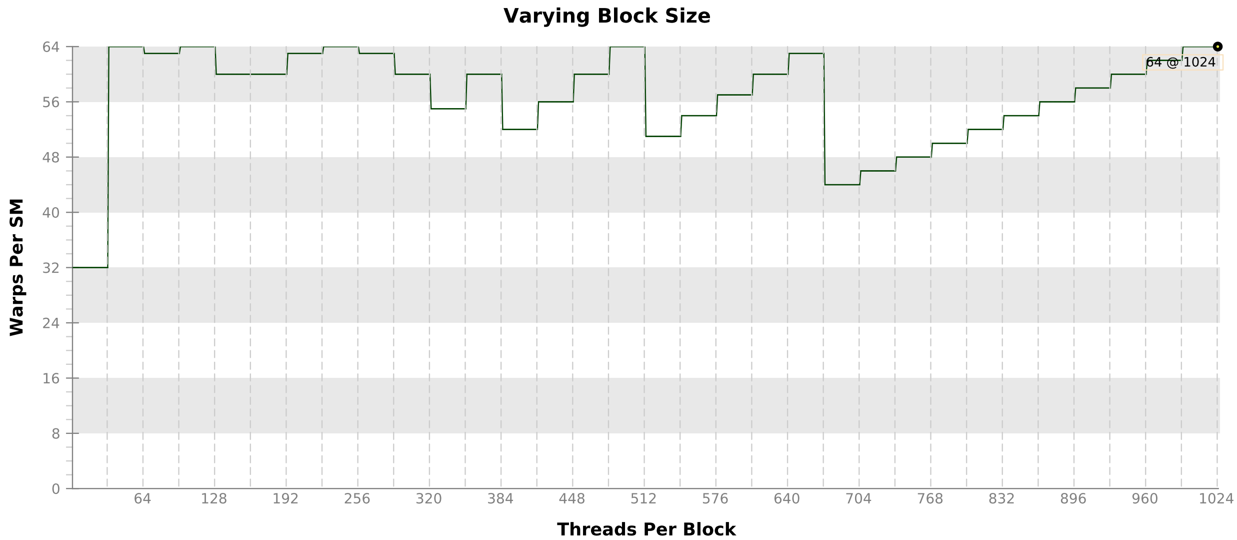
2.1. Occupancy Is Not Limiting Kernel Performance

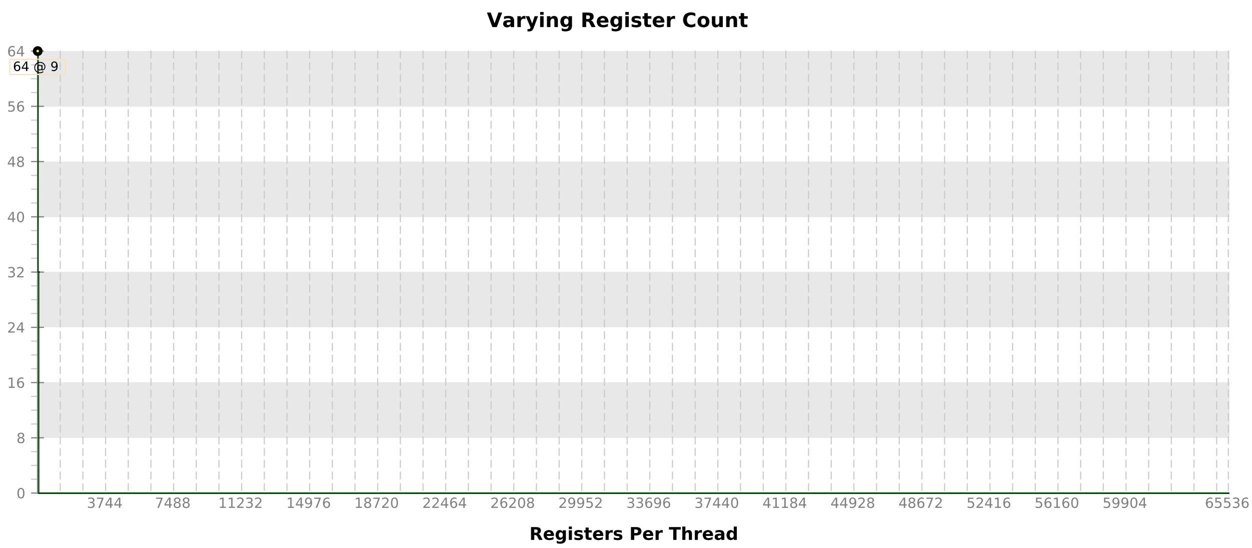
The kernel's block size, register usage, and shared memory usage allow it to fully utilize all warps on the GPU.

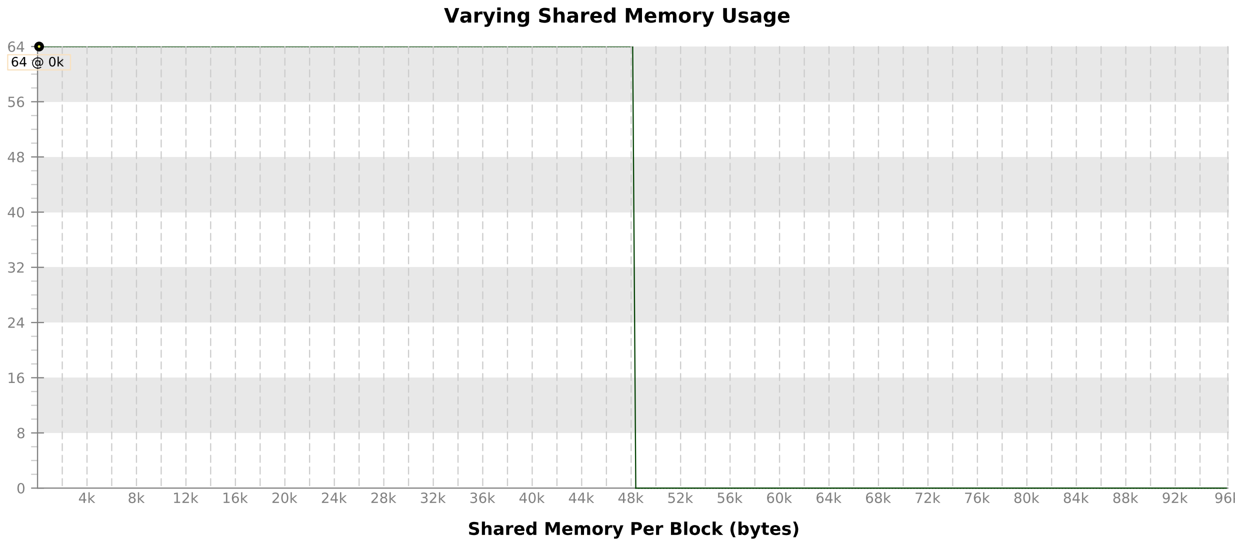
Variable	Achieved	Theoretical	Device Limit	Grid Size: [4,128,1] (512 blocks) Block Size: [32,32,1]
Occupancy Per SM				
Active Blocks		2	32	0 4 8 12 16 20 24 28 32
Active Warps	50	64	64	0 9 18 27 36 45 54 6364
Active Threads		2048	2048	0 512 1024 1536 2048
Occupancy	78.1%	100%	100%	0% 25% 50% 75% 100%
Warps				
Threads/Block		1024	1024	0 256 512 768 1024
Warps/Block		32	32	0 4 8 12 16 20 24 28 32
Block Limit		2	32	0 4 8 12 16 20 24 28 32
Registers				
Registers/Thread		9	65536	0 16384 32768 49152 65536
Registers/Block		16384	65536	0 16k 32k 48k 64k
Block Limit		4	32	0 4 8 12 16 20 24 28 32
Shared Memory				
Shared Memory/Block		0	98304	0 32k 64k 96k
Block Limit		0	32	0 4 8 12 16 20 24 28 32

2.2. Occupancy Charts

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

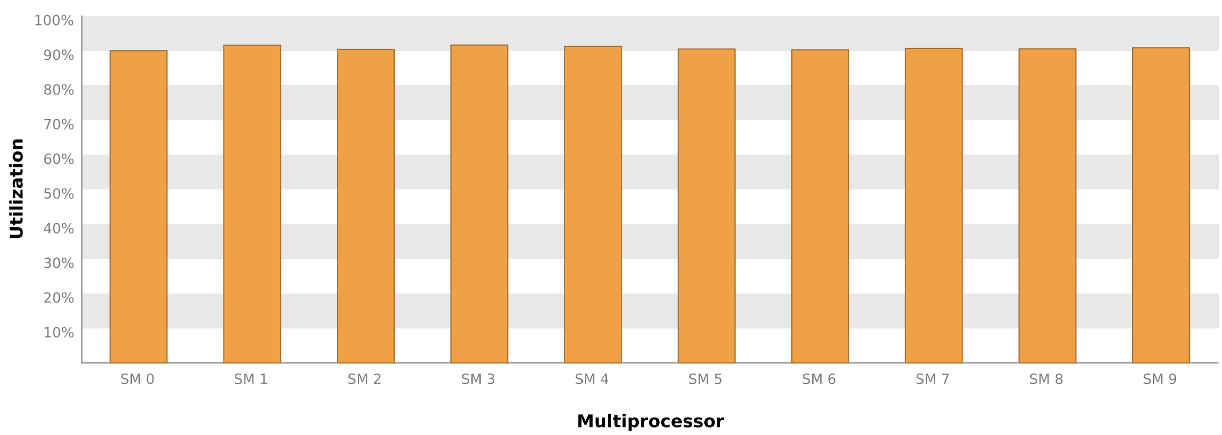

Varying Block Size
Warps Per SM
64
56
48
40
32
24
16
8
0
64
128
192
256
320
384
448
512
576
640
704
768
832
896
960
1024
64 @ 1024
Threads Per Block


Varying Register Count
64
56
48
40
32
24
16
8
0
64 @ 9
3744
7488
11232
14976
18720
22464
26208
29952
33696
37440
41184
44928
48672
52416
56160
59904
65536
Registers Per Thread



2.3. Multiprocessor Utilization

The kernel's blocks are distributed across the GPU's multiprocessors for execution. Depending on the number of blocks and the execution duration of each block some multiprocessors may be more highly utilized than others during execution of the kernel. The following chart shows the utilization of each multiprocessor during execution of the kernel.



3. 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 all threads in a warp have the same branching and predication behavior. The results below indicate that a significant fraction of the available compute performance is being wasted because branch and predication behavior is differing for threads within a warp.

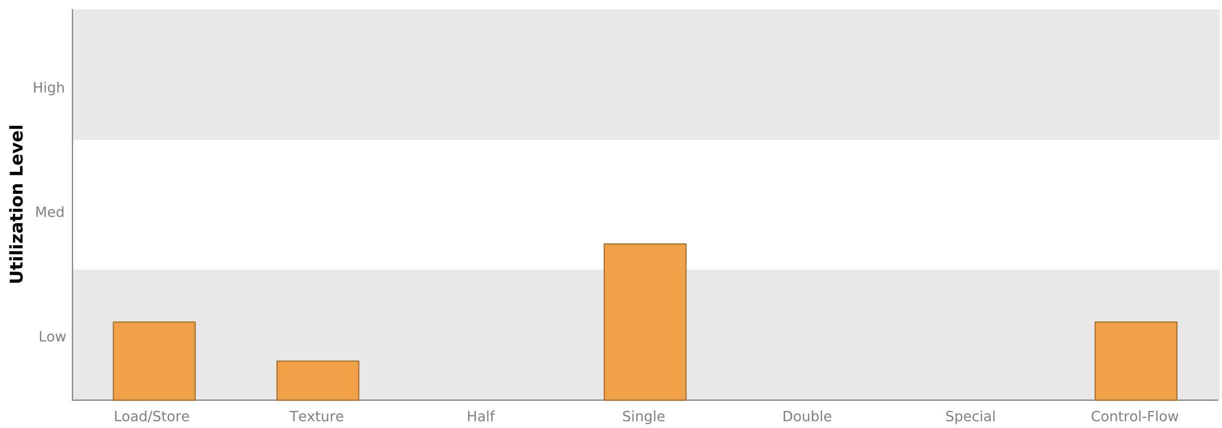
3.1. Divergent Branches

Compute resource are used most efficiently when all threads in a warp have the same branching behavior. When this does not occur the branch is said to be divergent. Divergent branches lower warp execution efficiency which leads to inefficient use of the GPU's compute resources.

Optimization: Each entry below points to a divergent branch within the kernel. For each branch reduce the amount of intra-warp divergence.

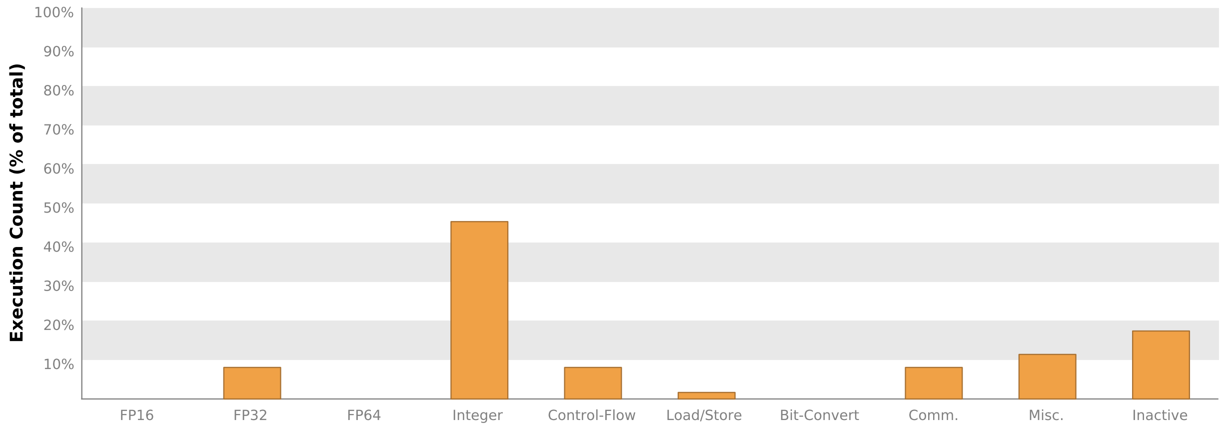
3.2. Function Unit Utilization

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 not limited by overuse of any function unit.
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.



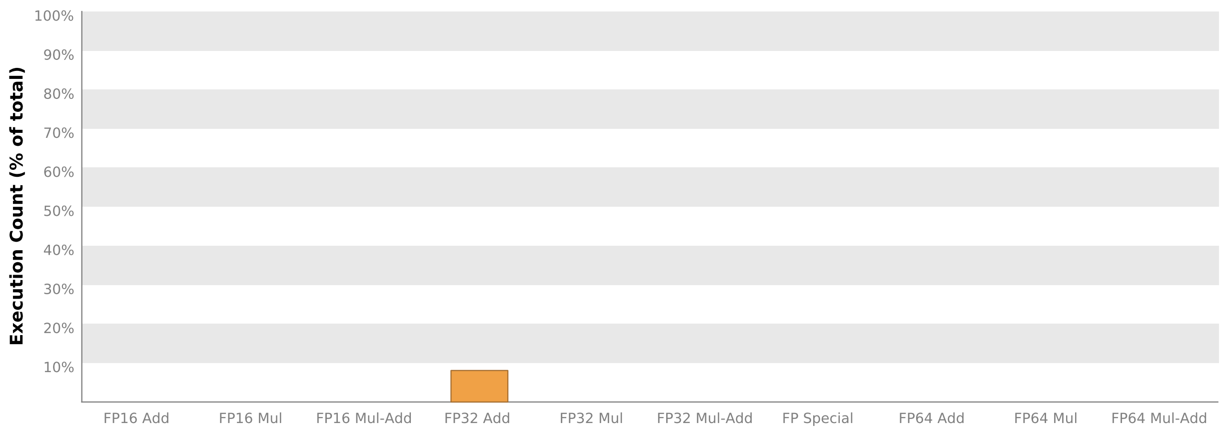
3.3. Instruction Execution Counts

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.



3.4. 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.



4. 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.

4.1. 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	
Shared Memory			
Shared Loads	0	0 B/s	
Shared Stores	0	0 B/s	
Shared Total	0	0 B/s	Idle Low Medium High Max
L2 Cache			
Reads	65568	64.651 GB/s	
Writes	16397	16.168 GB/s	
Total	81965	80.818 GB/s	Idle Low Medium High Max
Unified Cache			
Local Loads	0	0 B/s	
Local Stores	0	0 B/s	
Global Loads	131072	64.619 GB/s	
Global Stores	16384	16.155 GB/s	
Texture Reads	65536	64.619 GB/s	
Unified Total	212992	145.393 GB/s	Idle Low Medium High Max
Device Memory			
Reads	66356	65.428 GB/s	
Writes	22159	21.849 GB/s	
Total	88515	87.277 GB/s	Idle Low Medium High Max
System Memory			
[PCIe configuration: Gen2 x16, 5 Gbit/s]			
Reads	0	0 B/s	Idle Low Medium High Max
Writes	5	4.93 MB/s	Idle Low Medium High Max