

Analysis Report

norm(float*, float*, float*, int)

Duration	12.581 ms (12,580,928 ns)
Grid Size	[160,160,1]
Block Size	[16,16,1]
Registers/Thread	31
Shared Memory/Block	0 B
Shared Memory Requested	48 KiB
Shared Memory Executed	48 KiB
Shared Memory Bank Size	4 B

[0] GeForce GTX 480

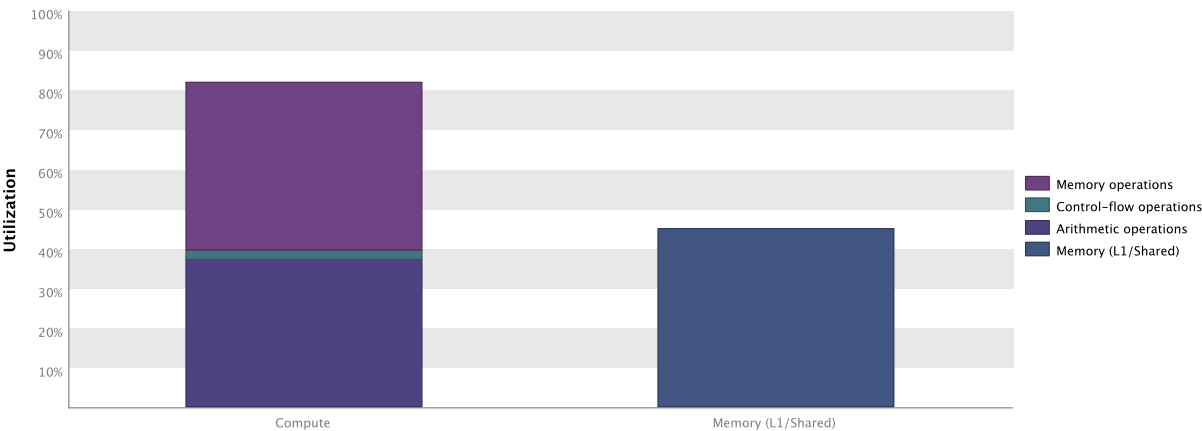
GPU UUID	GPU-e78c6efd-ec46-ea5f-d526-f6482d0c35f1
Compute Capability	2.0
Max. Threads per Block	1024
Max. Shared Memory per Block	48 KiB
Max. Registers per Block	32768
Max. Grid Dimensions	[65535, 65535, 65535]
Max. Block Dimensions	[1024, 1024, 64]
Max. Warps per Multiprocessor	48
Max. Blocks per Multiprocessor	8
Single Precision FLOP/s	672.48 GigaFLOP/s
Double Precision FLOP/s	336.24 GigaFLOP/s
Number of Multiprocessors	15
Multiprocessor Clock Rate	1.401 GHz
Concurrent Kernel	true
Max IPC	2
Threads per Warp	32
Global Memory Bandwidth	177.408 GB/s
Global Memory Size	1.5 GiB
Constant Memory Size	64 KiB
L2 Cache Size	768 KiB
Memcpy Engines	1
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 "norm" 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 480". 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




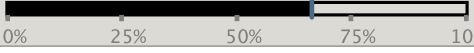
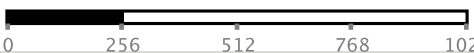
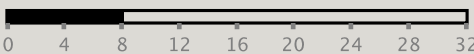



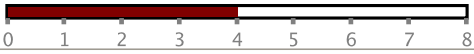

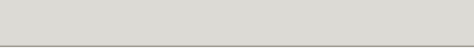
Instruction and memory latency limit the performance of a kernel when the GPU does not have enough work to keep busy. The performance of latency-limited kernels can often be improved by increasing occupancy. Occupancy is a measure of how many warps the kernel has active on the GPU, relative to the maximum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

2.1. GPU Utilization May Be Limited By Register Usage

Theoretical occupancy is less than 100% but is large enough that increasing occupancy may not improve performance. You can attempt the following optimization to increase the number of warps on each SM but it may not lead to increased performance.

The kernel uses 31 registers for each thread (7936 registers for each block). This register usage is likely preventing the kernel from fully utilizing the GPU. Device "GeForce GTX 480" provides up to 32768 registers for each block. Because the kernel uses 7936 registers for each block each SM is limited to simultaneously executing 4 blocks (32 warps). Chart "Varying Register Count" below shows how changing register usage will change the number of blocks that can execute on each SM.

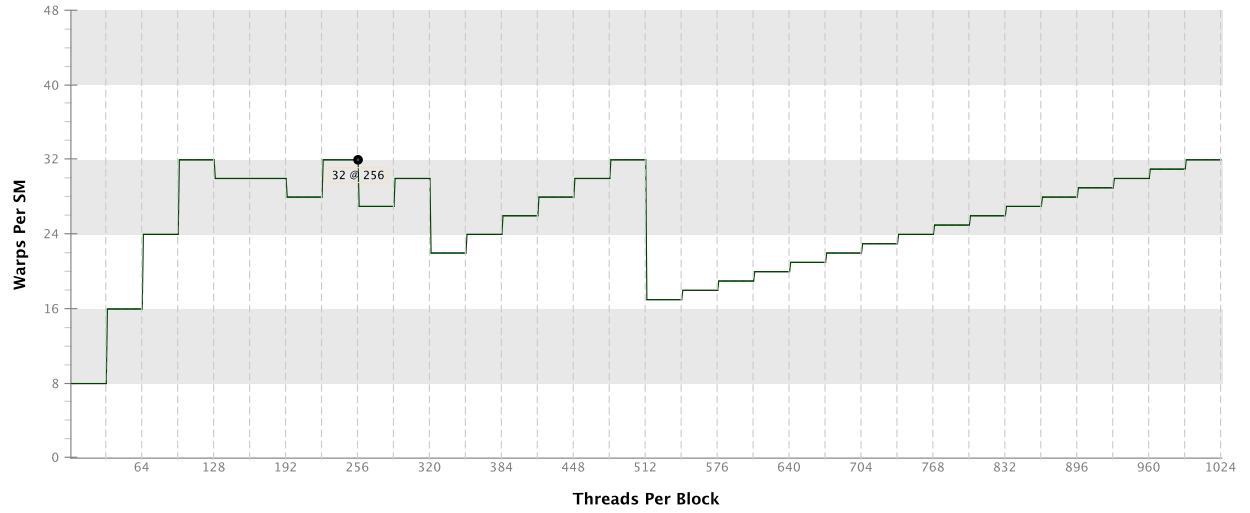
Optimization: Use the `-maxrregcount` flag or the `__launch_bounds__` qualifier to decrease the number of registers used by each thread. This will increase the number of blocks that can execute on each SM. On devices with Compute Capability 5.2 turning global cache off can increase the occupancy limited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

Variable	Achieved	Theoretical	Device Limit	Grid Size: [160,160,1] (25600 blocks) Block Size: [16,16]
Occupancy Per SM				
Active Blocks		4	8	
Active Warps	31.66	32	48	
Active Threads		1024	1536	
Occupancy	66%	66.7%	100%	
Warps				
Threads/Block		256	1024	
Warps/Block		8	32	
Block Limit		6	8	
Registers				
Registers/Thread		31	63	
Registers/Block		8192	32768	
Block Limit		4	8	
Shared Memory				
Shared Memory/Block		0	49152	
Block Limit			8	

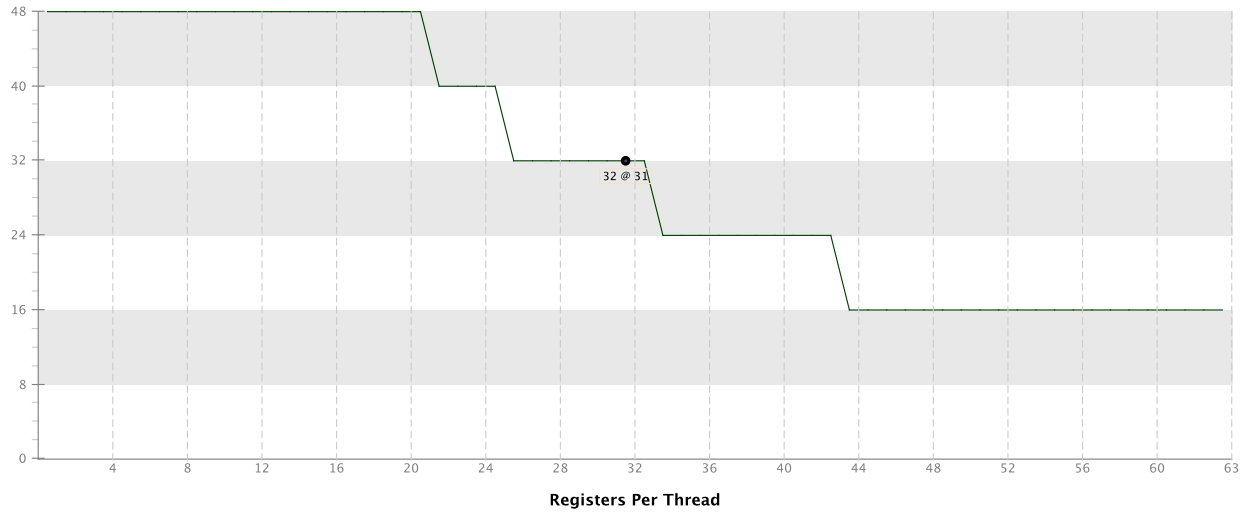
2.2. Occupancy Charts

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

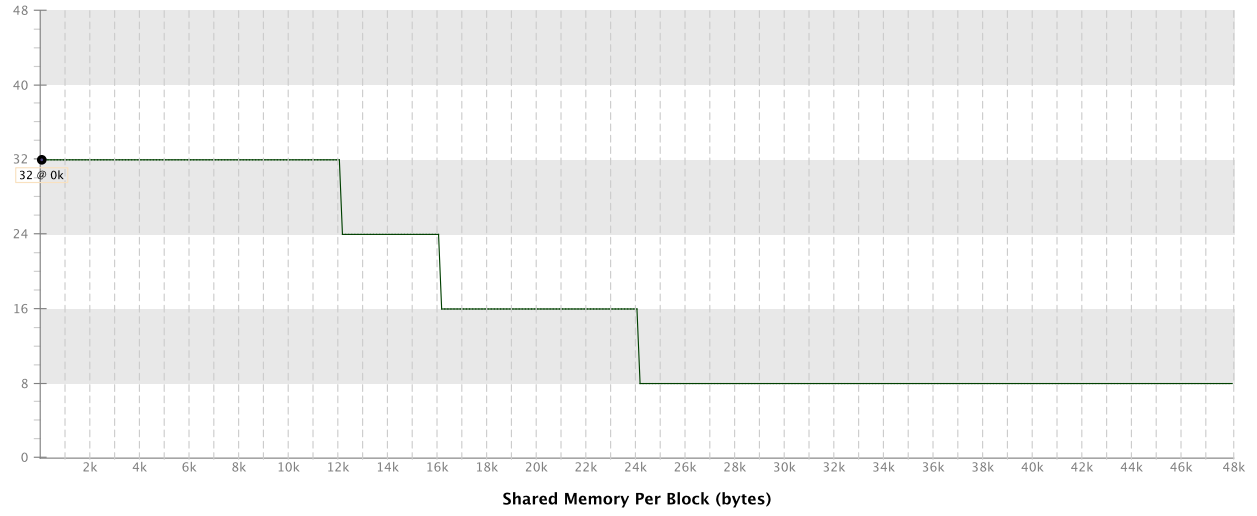
Varying Block Size



Varying Register Count



Varying Shared Memory Usage



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

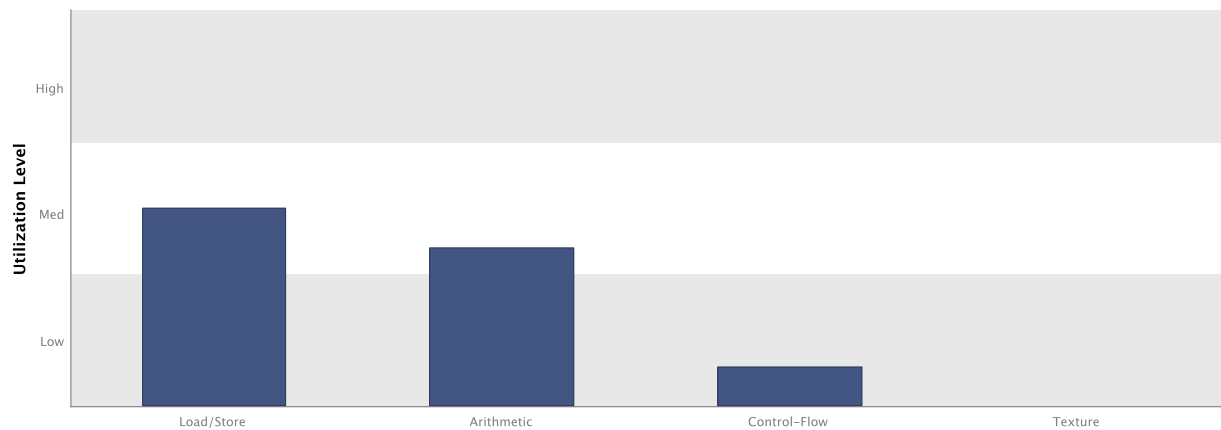
Compute resource are used most efficiently when all threads in a warp have the same branching behavior. When this does not occur the branch is said to be divergent. Divergent branches lower warp execution efficiency which leads to inefficient use of the GPU's compute resources.

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

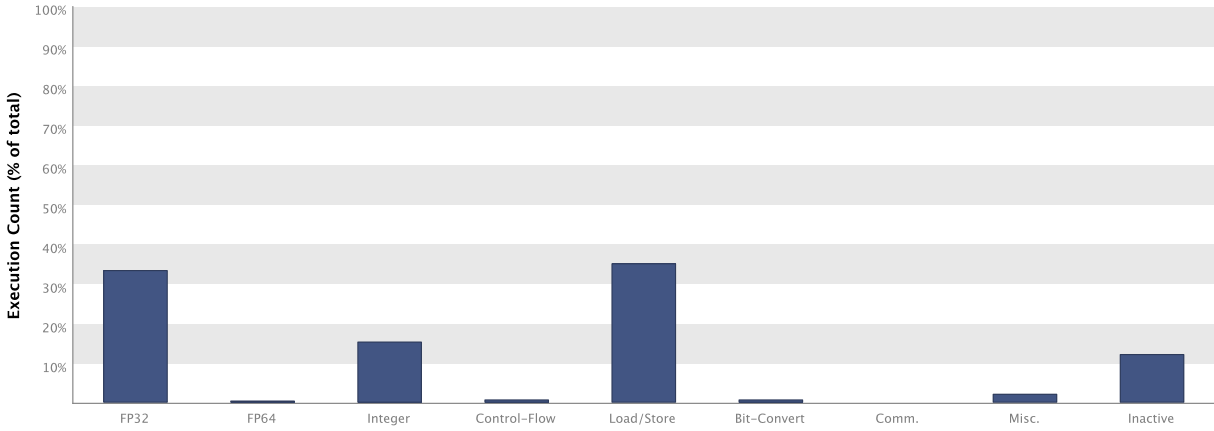
Different types of instructions are executed on different function units within each SM. Performance can be limited if a function unit is over-used by the instructions executed by the kernel. The following results show that the kernel's performance is not limited by overuse of any function unit.

- Load/Store - Load and store instructions for 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.



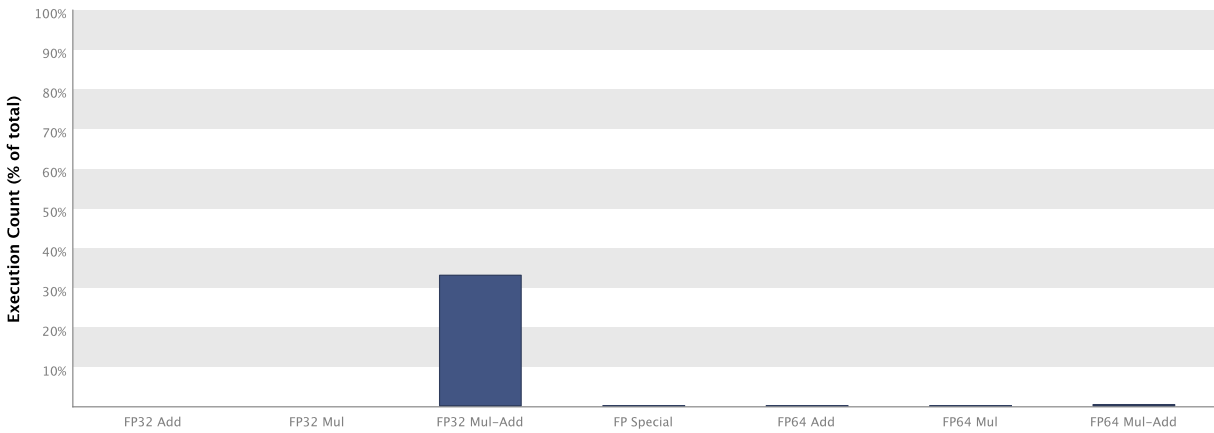
3.3. Instruction Execution Counts

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
L1/Shared Memory			
Local Loads	0	0 B/s	
Local Stores	0	0 B/s	
Shared Loads	0	0 B/s	
Shared Stores	0	0 B/s	
Global Loads	60623760	625.96 GB/s	
Global Stores	6553920	16.917 GB/s	
Atomic	0	0 B/s	
L1/Shared Total	67177680	642.877 GB/s	
L2 Cache			
L1 Reads	1650524	4.261 GB/s	
L1 Writes	6553600	16.917 GB/s	
Texture Reads	0	0 B/s	
Atomic	0	0 B/s	
Total	8204124	21.178 GB/s	
Texture Cache			
Reads	0	0 B/s	
Device Memory			
Reads	819502	2.115 GB/s	
Writes	3260752	8.417 GB/s	
Total	4080254	10.532 GB/s	
System Memory			
[PCIe configuration: Gen2 x16, 5 Gbit/s]			
Reads	0	0 B/s	
Writes	0	0 B/s	