Analysis Report

void computeHOGSharedPred<float, int=8, int=8, int=16, int=16, int=64>(float const *, float const *, float*, float const *, int, int, int, int)

Duration	168.451 µs
Grid Size	[6,1,1]
Block Size	[256,1,1]
Registers/Thread	40
Shared Memory/Block	36 KiB
Shared Memory Requested	96 KiB
Shared Memory Executed	96 KiB
Shared Memory Bank Size	4 B

[0] GeForce GTX 960

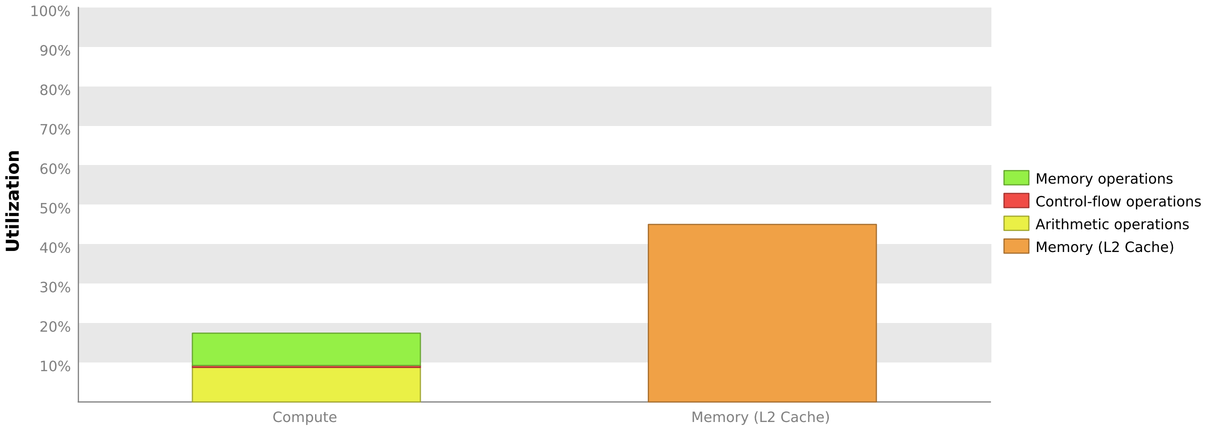
GPU UUID	GPU-0db32734-f94e-48a7-8b5d-4604317dc554
Compute Capability	5.2
Max. Threads per Block	1024
Max. Shared Memory per Block	48 KiB
Max. Registers per Block	65536
Max. Grid Dimensions	[2147483647, 65535, 65535]
Max. Block Dimensions	[1024, 1024, 64]
Max. Warps per Multiprocessor	64
Max. Blocks per Multiprocessor	32
Single Precision FLOP/s	2.644 TeraFLOP/s
Double Precision FLOP/s	82.624 GigaFLOP/s
Number of Multiprocessors	8
Multiprocessor Clock Rate	1.291 GHz
Concurrent Kernel	true
Max IPC	6
Threads per Warp	32
Global Memory Bandwidth	112.16 GB/s
Global Memory Size	4 GiB
Constant Memory Size	64 KiB
L2 Cache Size	1 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 "void computeHOGSharedPred<f..." 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 960". 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.



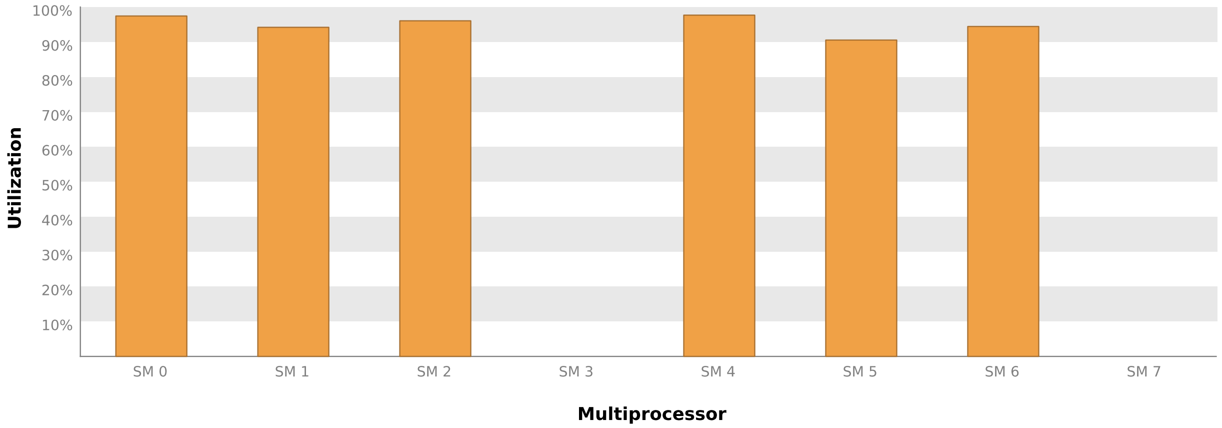
2. Instruction and Memory Latency

Instruction and memory latency limit the performance of a kernel when the GPU does not have enough work to keep busy. The results below indicate that the GPU does not have enough work because differences in the execution time of the kernel's blocks leads to poor load balancing across the SMs.

2.1. Achieved Occupancy Is Low

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 kernel's achieved occupancy of 11.8% is significantly lower than its theoretical occupancy of 25%. Most likely this indicates that there is an imbalance in how the kernel's blocks are executing on the SMs so that all SMs are not equally busy over the entire execution of the kernel. The following chart shows the utilization of each multiprocessor during execution of the kernel.

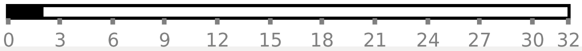
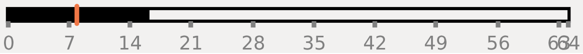
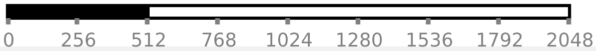
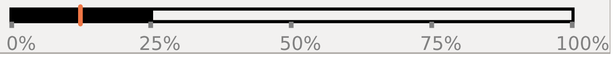
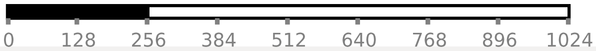
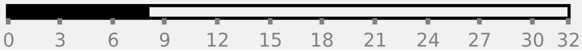
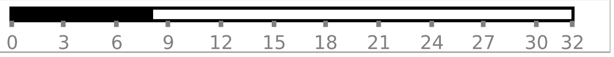
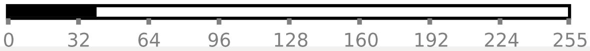
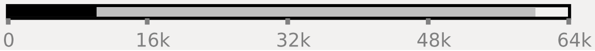
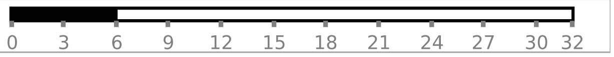
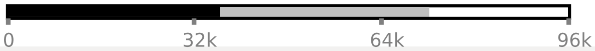
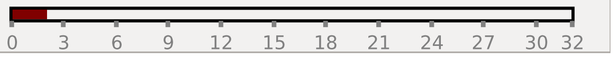
Optimization: Make sure that all blocks are doing roughly the same amount of work. It may also help to increase the number of blocks executed by the kernel.



2.2. GPU Utilization Is Limited By Shared Memory Usage

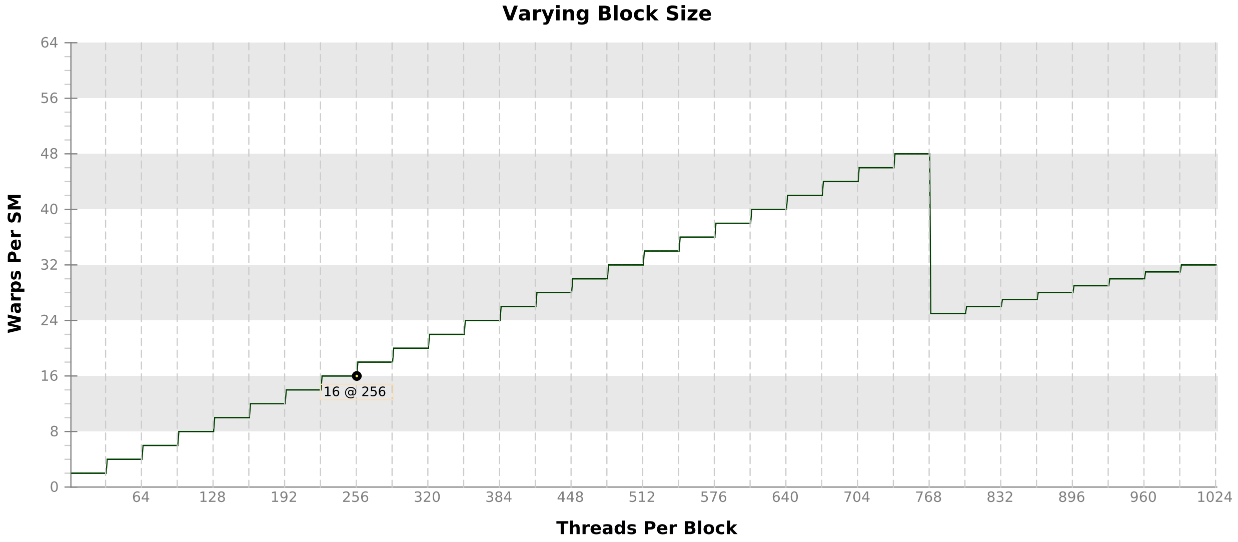
The kernel uses 36 KiB of shared memory for each block. This shared memory usage is likely preventing the kernel from fully utilizing the GPU. Device "GeForce GTX 960" is configured to have 96 KiB of shared memory for each SM. Because the kernel uses 36 KiB of shared memory for each block each SM is limited to simultaneously executing 2 blocks (16 warps). Chart "Varying Shared Memory Usage" below shows how changing shared memory usage will change the number of blocks that can execute on each SM.

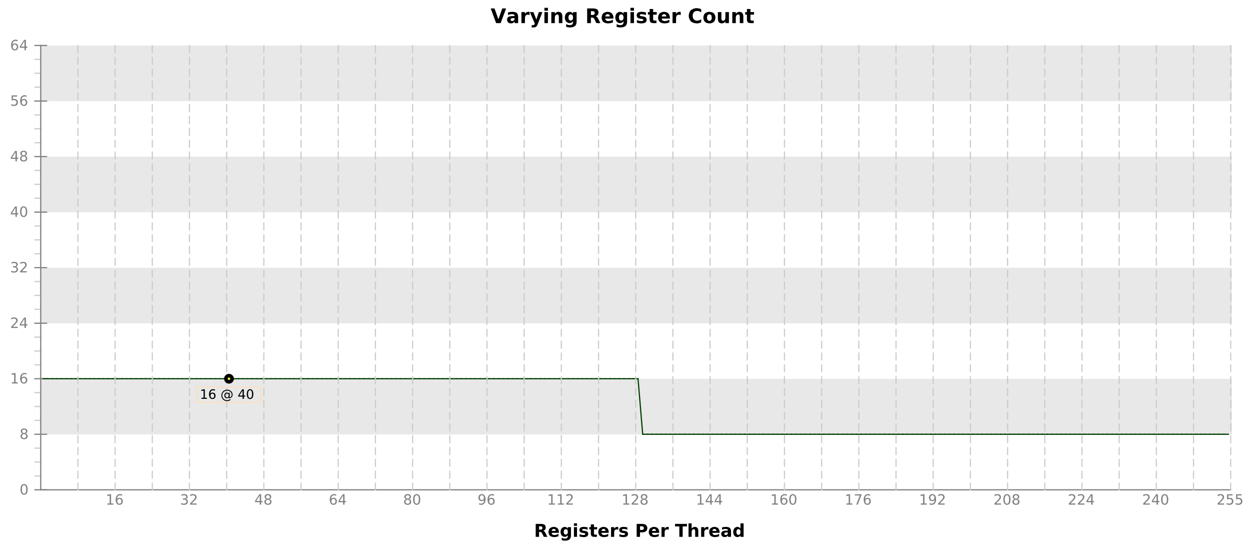
Optimization: Reduce shared memory usage to increase the number of blocks that can execute on each SM. You can also increase the number of blocks that can execute on each SM by increasing the amount of shared memory available to your kernel. You do this by setting the preferred cache configuration to "prefer shared".

Variable	Achieved	Theoretical	Device Limit	Grid Size: [6,1,1] (6 blocks) Block Size: [256,1,1] (256 threads)
Occupancy Per SM				
Active Blocks		2	32	
Active Warps	7.55	16	64	
Active Threads		512	2048	
Occupancy	11.8%	25%	100%	
Warps				
Threads/Block		256	1024	
Warps/Block		8	32	
Block Limit		8	32	
Registers				
Registers/Thread		40	255	
Registers/Block		10240	65536	
Block Limit		6	32	
Shared Memory				
Shared Memory/Block		36864	98304	
Block Limit		2	32	

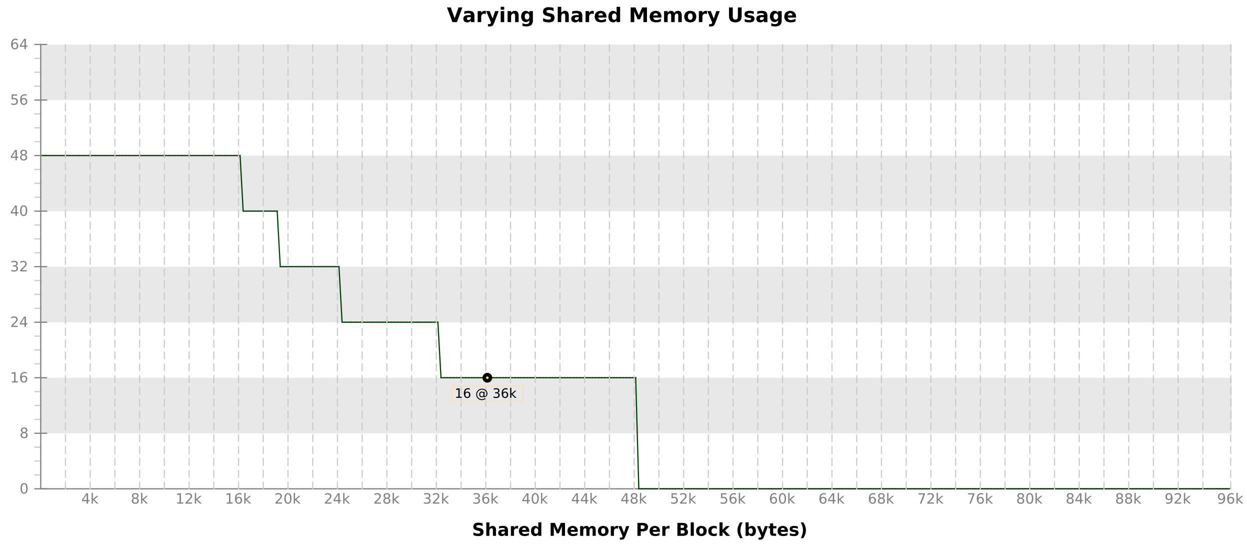
2.3. Occupancy Charts

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





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.

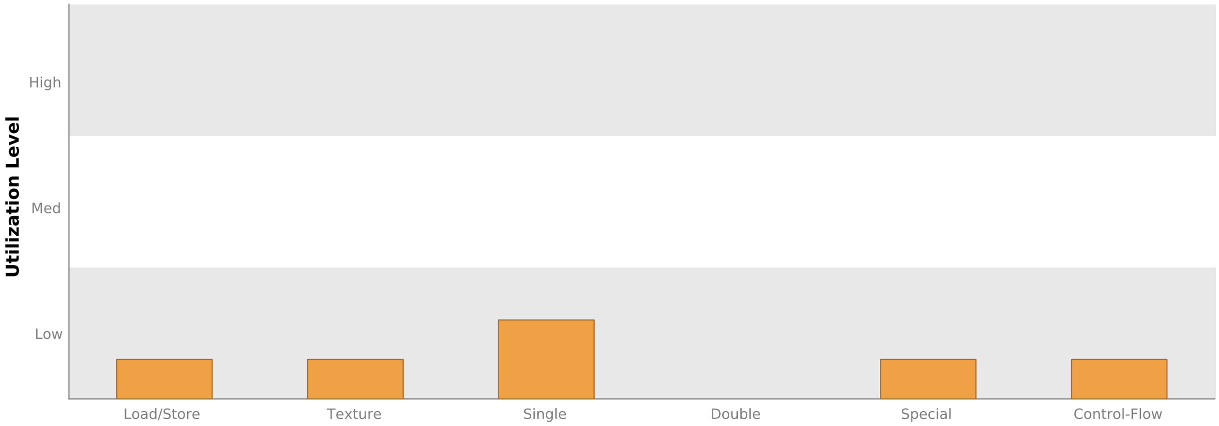
Optimization: Each entry below points to a divergent branch within the kernel. For each branch reduce the amount of intra-warp divergence.

	/home/adas/cuda-workspace/CudaVisionSysDeploy/Release/../src/init/../device/HOG/HOGdescriptor.cuh
Line 146	Divergence = 2.1% [1 divergent executions out of 48 total executions]

3.2. Function Unit Utilization

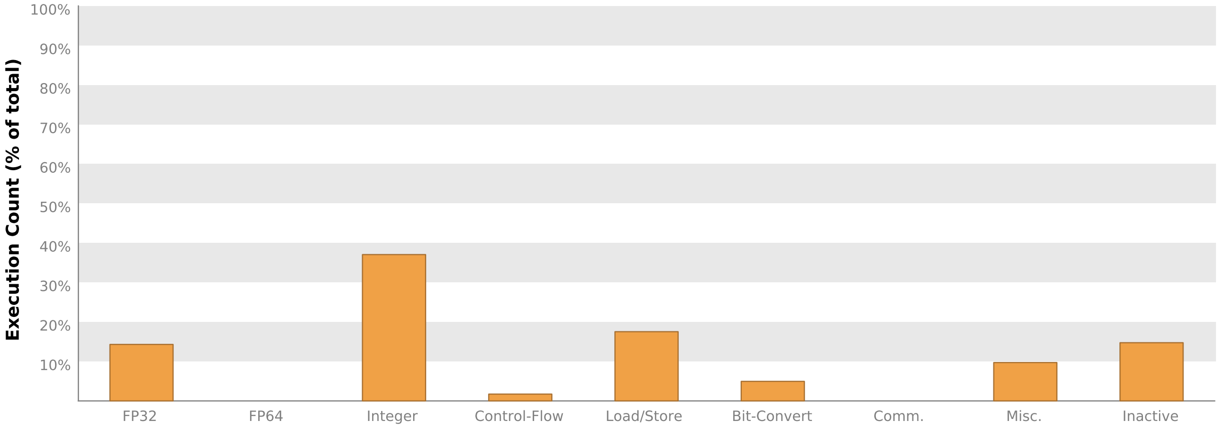
Different types of instructions are executed on different function units within each SM. Performance can be limited if a function unit is over-used by the instructions executed by the kernel. The following results show that the kernel's performance is not limited by overuse of any function unit.

Load/Store - Load and store instructions for shared and constant memory.
Texture - Load and store instructions for local, global, and texture memory.
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.



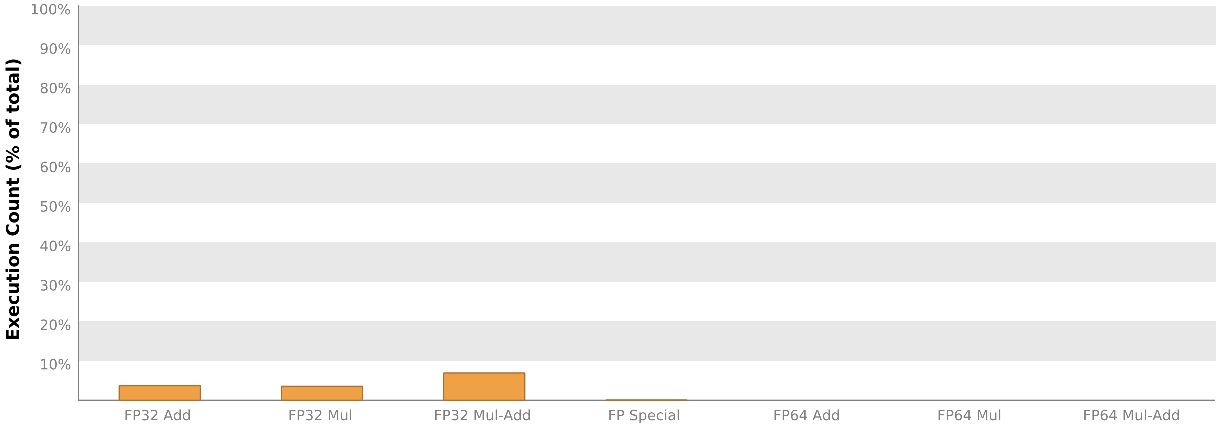
3.3. Instruction Execution Counts

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.

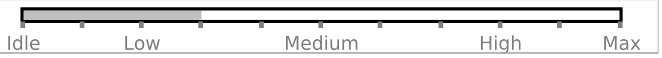
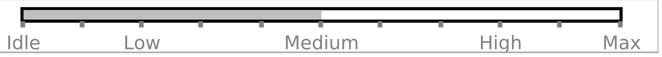
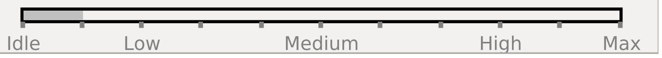
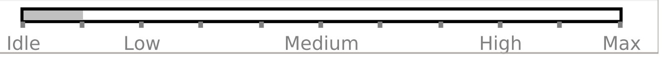
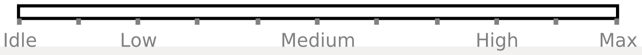
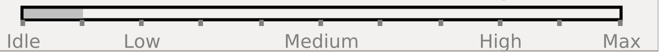


4. Memory Bandwidth

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	230528	171.598 GB/s	
Shared Stores	230888	171.866 GB/s	
Shared Total	461416	343.463 GB/s	
L2 Cache			
Reads	746220	138.866 GB/s	
Writes	52242	9.722 GB/s	
Total	798462	148.587 GB/s	
Unified Cache			
Local Loads	0	0 B/s	
Local Stores	0	0 B/s	
Global Loads	868128	161.552 GB/s	
Global Stores	52236	9.721 GB/s	
Texture Reads	218400	40.642 GB/s	
Unified Total	1138764	211.915 GB/s	
Device Memory			
Reads	22803	4.243 GB/s	
Writes	16036	2.984 GB/s	
Total	38839	7.228 GB/s	
System Memory			
[PCIe configuration: Gen2 x16, 5 Gbit/s]			
Reads	0	0 B/s	
Writes	5	930.459 kB/s	 Idle Low Medium High Max