Analysis Report

void mergeHistosSIMDaccum<float, unsigned int=64>(unsigned char*, unsigned char*, float*, unsigned int, unsigned int)

Duration	340.5 µs
Grid Size	[71,1,1]
Block Size	[256,1,1]
Registers/Thread	30
Shared Memory/Block	0 B
Shared Memory Requested	96 KiB
Shared Memory Executed	96 KiB
Shared Memory Bank Size	4 B

[0] GeForce GTX 960

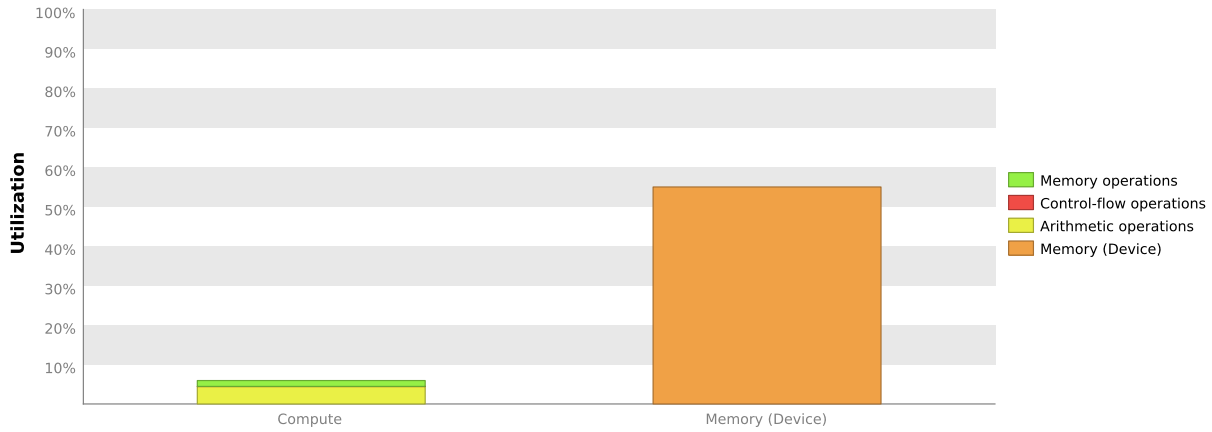
GPU UUID	GPU-0db32734-f94e-48a7-8b5d-4604317dc554
Compute Capability	5.2
Max. Threads per Block	1024
Max. Shared Memory per Block	48 KiB
Max. Registers per Block	65536
Max. Grid Dimensions	[2147483647, 65535, 65535]
Max. Block Dimensions	[1024, 1024, 64]
Max. Warps per Multiprocessor	64
Max. Blocks per 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 mergeHistosSIMDaccum<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 instruction execution is stalling excessively.

2.1. Kernel Profile - PC Sampling

The Kernel Profile - PC Sampling gives the number of samples for each source and assembly line with various stall reasons. Using this information you can pinpoint portions of your kernel that are introducing latencies and the reason for the latency. Samples are taken in round robin order for all active warps at a fixed number of cycles regardless of whether the warp is issuing an instruction or not.

Instruction Issued - Warp was issued

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.

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.

Texture - The texture sub-system is fully utilized or has too many outstanding requests.

Synchronization - The warp is blocked at a __syncthreads() call.

Constant - A constant load is blocked due to a miss in the constants cache.

Pipe Busy - The compute resource(s) required by the instruction is not yet available.

Memory Throttle - Large number of pending memory operations prevent further forward progress. These can be reduced by combining several memory transactions into one.

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.

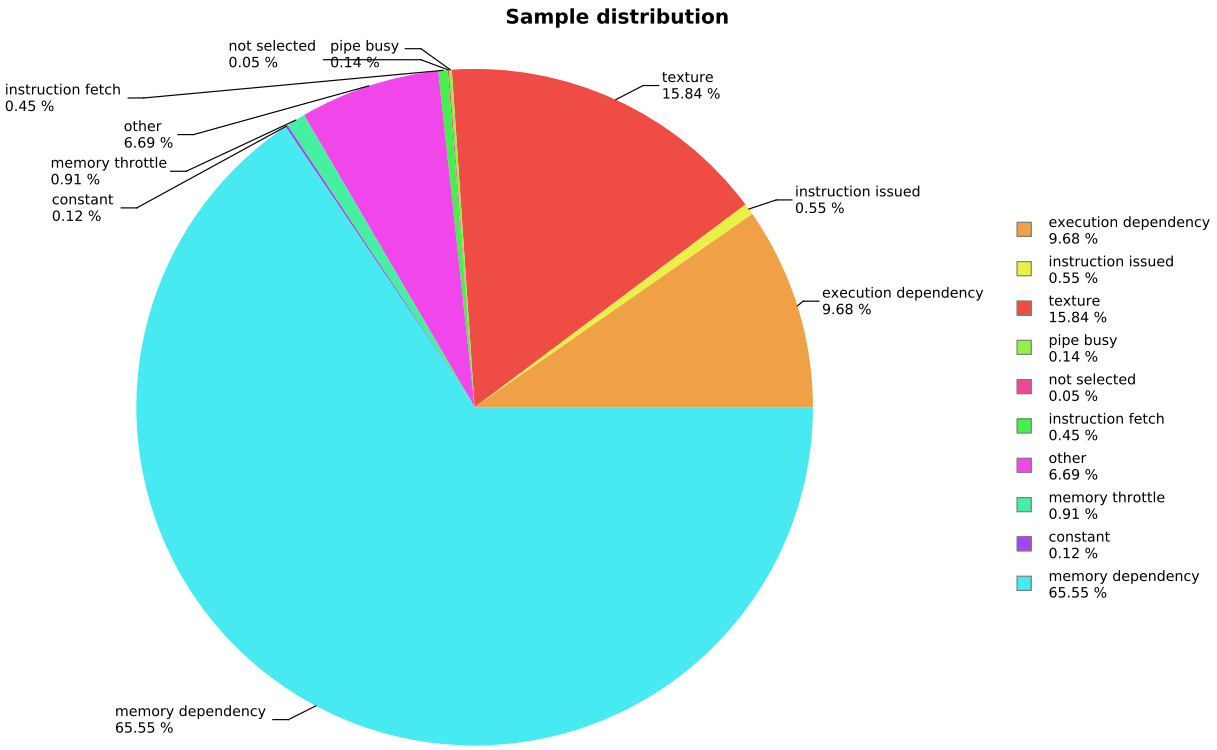
Other - The warp is blocked for a uncommon reason.

Examine portions of the kernel that have high number of samples to know where the maximum time was spent and observe the latency reasons for those samples to identify optimization opportunities.

Cuda Functions	Sample Count	% of Kernel Samples
void mergeHistosSIMDaccum<float, unsigned int=64>(unsigned char*, unsigned char*, float*, unsigned int, unsigned int)	23855	100.0

Source Files :

/home/adas/cuda-workspace/CudaVisionSysDeploy/Release/../src/init/../device/LBPHist/../Operations/simd_functions.h
/home/adas/cuda-workspace/CudaVisionSysDeploy/Release/../src/init/../device/LBPHist/blockHistograms.h



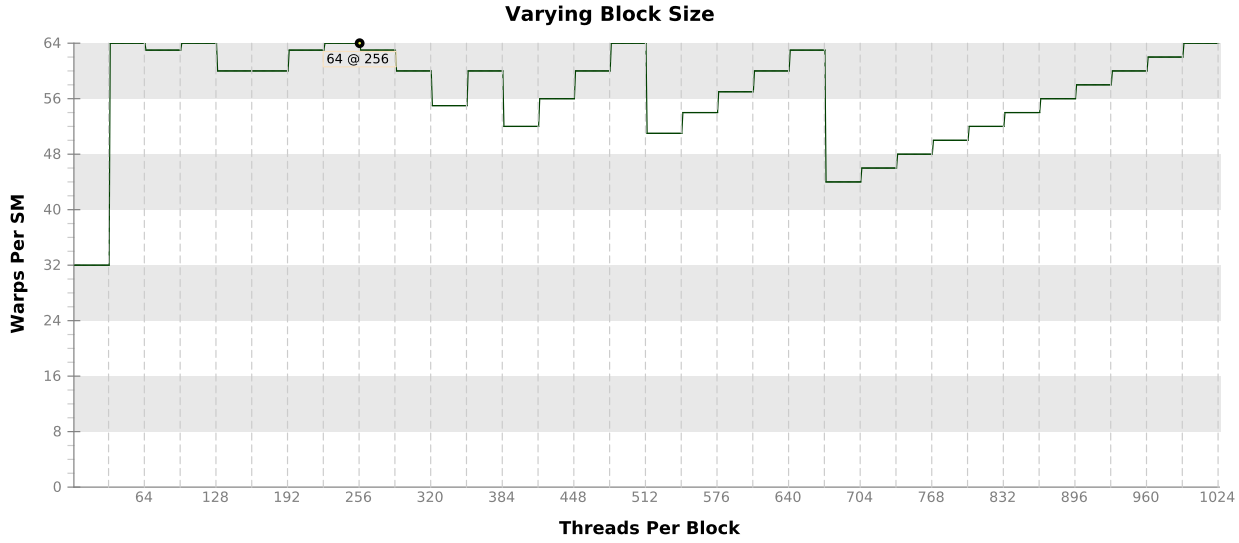
2.2. Occupancy Is Not Limiting Kernel Performance

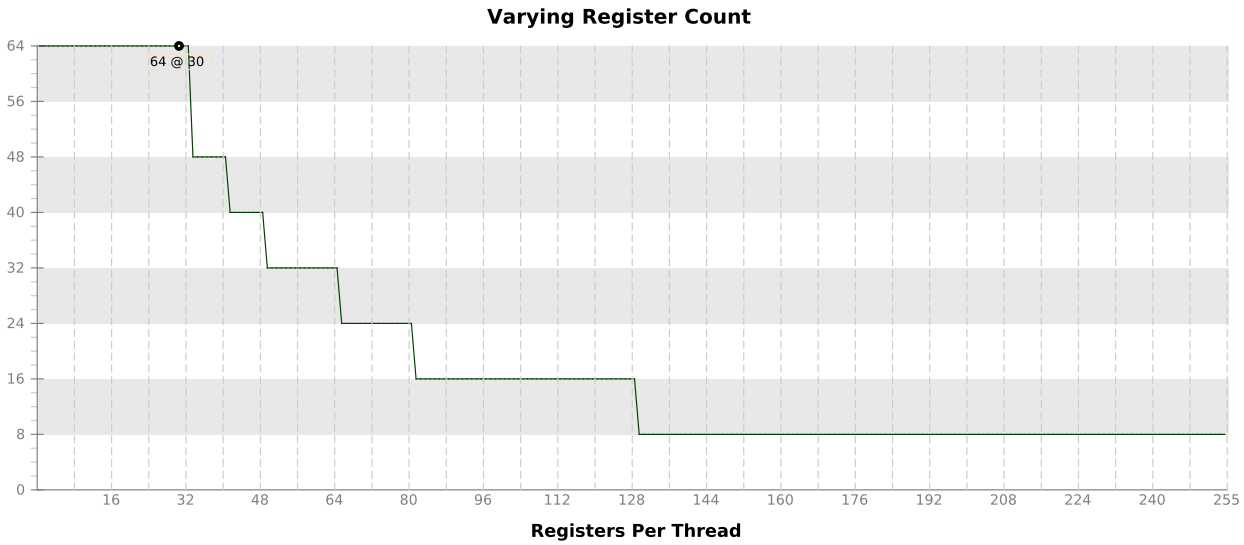
The kernel's block size, register usage, and shared memory usage allow it to fully utilize all warps on the GPU.

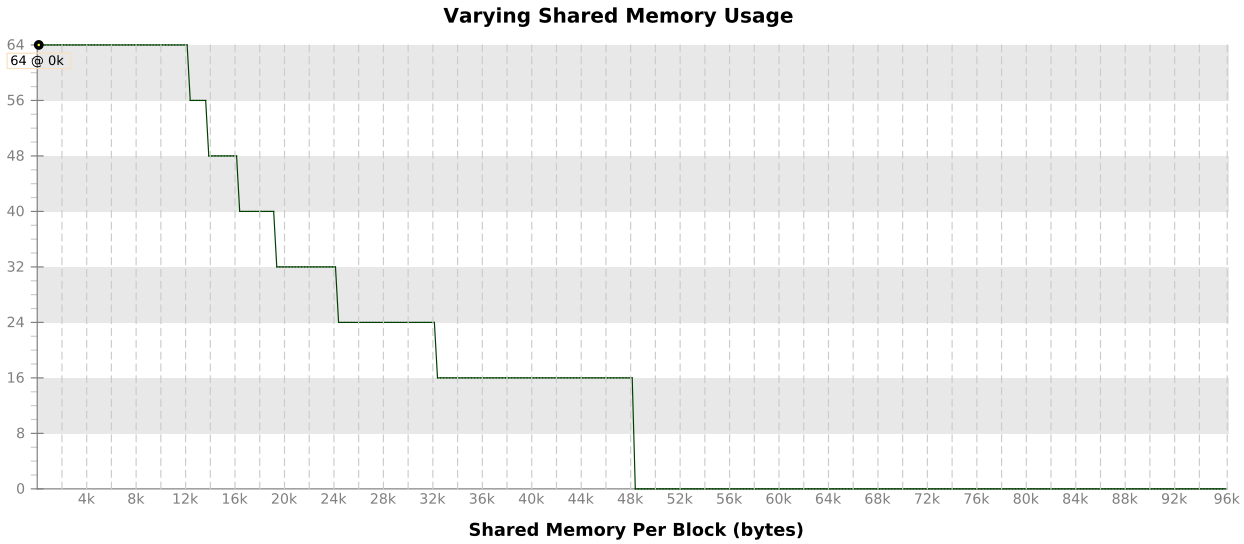
Variable	Achieved	Theoretical	Device Limit	Grid Size: [71,1,1] (71 blocks) Block Size: [256,1,1] (256 threads
Occupancy Per SM				
Active Blocks		8	32	0 3 6 9 12 15 18 21 24 27 30 32
Active Warps	61.34	64	64	0 7 14 21 28 35 42 49 56 64
Active Threads		2048	2048	0 256 512 768 1024 1280 1536 1792 2048
Occupancy	95.8%	100%	100%	0% 25% 50% 75% 100%
Warps				
Threads/Block		256	1024	0 128 256 384 512 640 768 896 1024
Warps/Block		8	32	0 3 6 9 12 15 18 21 24 27 30 32
Block Limit		8	32	0 3 6 9 12 15 18 21 24 27 30 32
Registers				
Registers/Thread		30	255	0 32 64 96 128 160 192 224 255
Registers/Block		8192	65536	0 16k 32k 48k 64k
Block Limit		8	32	0 3 6 9 12 15 18 21 24 27 30 32
Shared Memory				
Shared Memory/Block		0	98304	0 32k 64k 96k
Block Limit			32	

2.3. Occupancy Charts

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




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


Varying Shared Memory Usage
64 @ 0k
64
56
48
40
32
24
16
8
0
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)

3. Compute Resources

GPU compute resources limit the performance of a kernel when those resources are insufficient or poorly utilized.

3.1. Kernel Profile - Instruction Execution

The Kernel Profile - Instruction Execution shows the execution count, inactive threads, and predicated threads for each source and assembly line of the kernel. Using this information you can pinpoint portions of your kernel that are making inefficient use of compute resource due to divergence and predication.

Examine portions of the kernel that have high execution counts and inactive or predicated threads to identify optimization opportunities.

Cuda Fuctions :

void mergeHistosSIMDaccum<float, unsigned int=64>(unsigned char*, unsigned char*, float*, unsigned int, unsigned int)

Maximum instruction execution count in assembly: 2236
Average instruction execution count in assembly: 2115
Instructions executed for the kernel: 939246
Thread instructions executed for the kernel: 30034214
Non-predicated thread instructions executed for the kernel: 29998464
Warp non-predicated execution efficiency of the kernel: 99.8%
Warp execution efficiency of the kernel: 99.9%

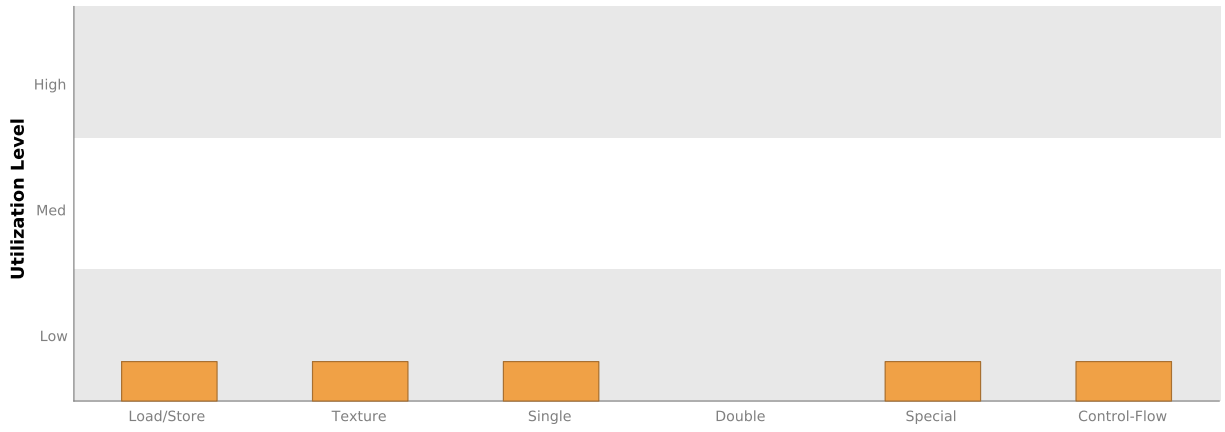
Source files :

/home/adas/cuda-workspace/CudaVisionSysDeploy/Release/../src/init/../device/LBPHist/../Operations/simd_functions.h
/home/adas/cuda-workspace/CudaVisionSysDeploy/Release/../src/init/../device/LBPHist/blockHistograms.h

3.2. Function Unit Utilization

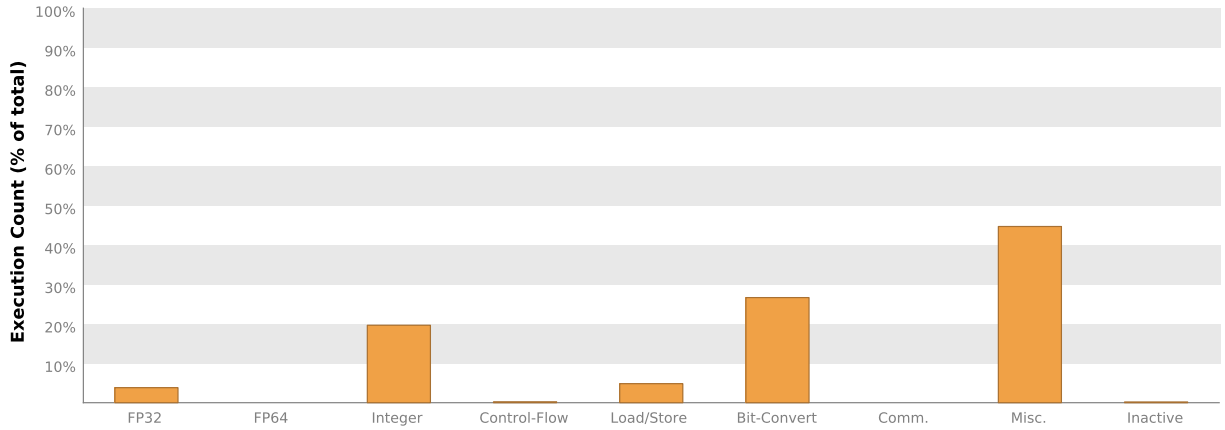
Different types of instructions are executed on different function units within each SM. Performance can be limited if a function unit is over-used by the instructions executed by the kernel. The following results show that the kernel's performance is not limited by overuse of any function unit.

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.



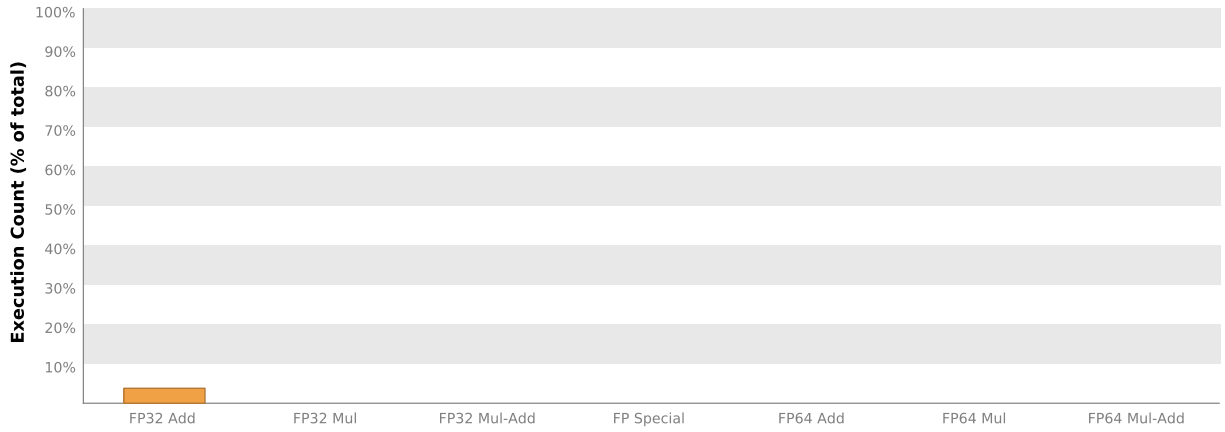
3.3. Instruction Execution Counts

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. The results below indicate that the kernel is limited by the bandwidth available to the device memory.

4.1. Global Memory Alignment and Access Pattern

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

Optimization: Each entry below points to a global 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.

/home/adas/cuda-workspace/CudaVisionSysDeploy/Release/../src/init/../device/LBPHist/blockHistograms.h

Line 76	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 76	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 76	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 76	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 76	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 76	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 76	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 76	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 79	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 79	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 79	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 79	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 79	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 79	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 79	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 79	Global Load L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 84	Global Store L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total

/home/adas/cuda-workspace/CudaVisionSysDeploy/Release/../src/init/../device/LBPHist/blockHistograms.h

	executions]
Line 84	Global Store L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 84	Global Store L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]
Line 84	Global Store L2 Transactions/Access = 32, Ideal Transactions/Access = 4 [71500 L2 transactions for 2236 total executions]

4.2. 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	0	0 B/s	
Shared Stores	0	0 B/s	
Shared Total	0	0 B/s	Idle Low Medium High Max
L2 Cache			
Reads	1144375	107.548 GB/s	
Writes	288241	27.089 GB/s	
Total	1432616	134.636 GB/s	Idle Low Medium High Max
Unified Cache			
Local Loads	0	0 B/s	
Local Stores	0	0 B/s	
Global Loads	1144000	107.512 GB/s	
Global Stores	288235	27.088 GB/s	
Texture Reads	143040	13.443 GB/s	
Unified Total	1575275	148.043 GB/s	Idle Low Medium High Max
Device Memory			
Reads	401159	37.701 GB/s	
Writes	268422	25.226 GB/s	
Total	669581	62.927 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	469.897 kB/s	Idle Low Medium High Max