Result	Size	Time	Cycles	GPU	SM Frequency	Process	Attributes
Current 788 - ob1_redu	(256, 1, 1)x(256, 1, 1)	10.50 us	17,113	0 - NVIDIA GeForce RTX 2080 SUPER	1.62 Ghz	[16536] benchmark_reduce.exe	

Summary | Details | Source | Context | Comments | Raw | Session — Compare | Tools | View | Export

GPU Speed Of Light Throughput

GPU Throughput Chart

High-level overview of the throughput for compute and memory resources of the GPU. For each unit, the throughput reports the achieved percentage of utilization with respect to the theoretical maximum. Breakdowns show the throughput for each individual sub-metric of Compute and Memory to clearly identify the highest contributor. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.

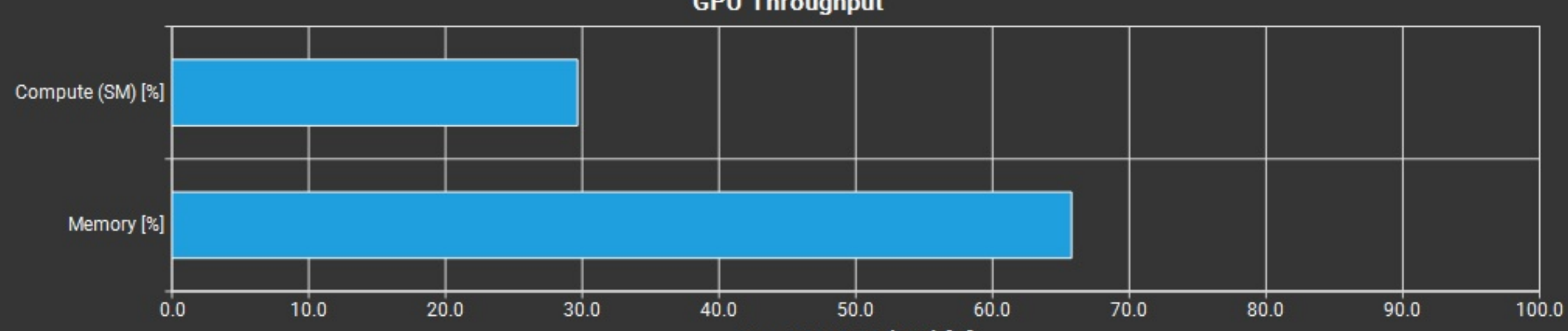
Compute (SM) Throughput [%]	29.66	Duration [us]	10.50
Memory Throughput [%]	65.78	Elapsed Cycles [cycle]	17,113
L1/TEX Cache Throughput [%]	15.10	SM Active Cycles [cycle]	16,705.04
L2 Cache Throughput [%]	19.64	SM Frequency [Ghz]	1.62
DRAM Throughput [%]	65.78	DRAM Frequency [Ghz]	7.56

High Memory Throughput Memory is more heavily utilized than Compute: Look at the Memory Workload Analysis section to identify the DRAM bottleneck. Check memory replay (coalescing) metrics to make sure you're efficiently utilizing the bytes transferred. Also consider whether it is possible to do more work per memory access (kernel fusion) or whether there are values you can (re)compute.

Key Performance Indicators

Roofline Analysis The ratio of peak float (FP32) to double (FP64) performance on this device is 32:1. The workload achieved 3% of this device's FP32 peak performance and 0% of its FP64 peak performance. See the Profiling Guide for more details on roofline analysis.

GPU Throughput

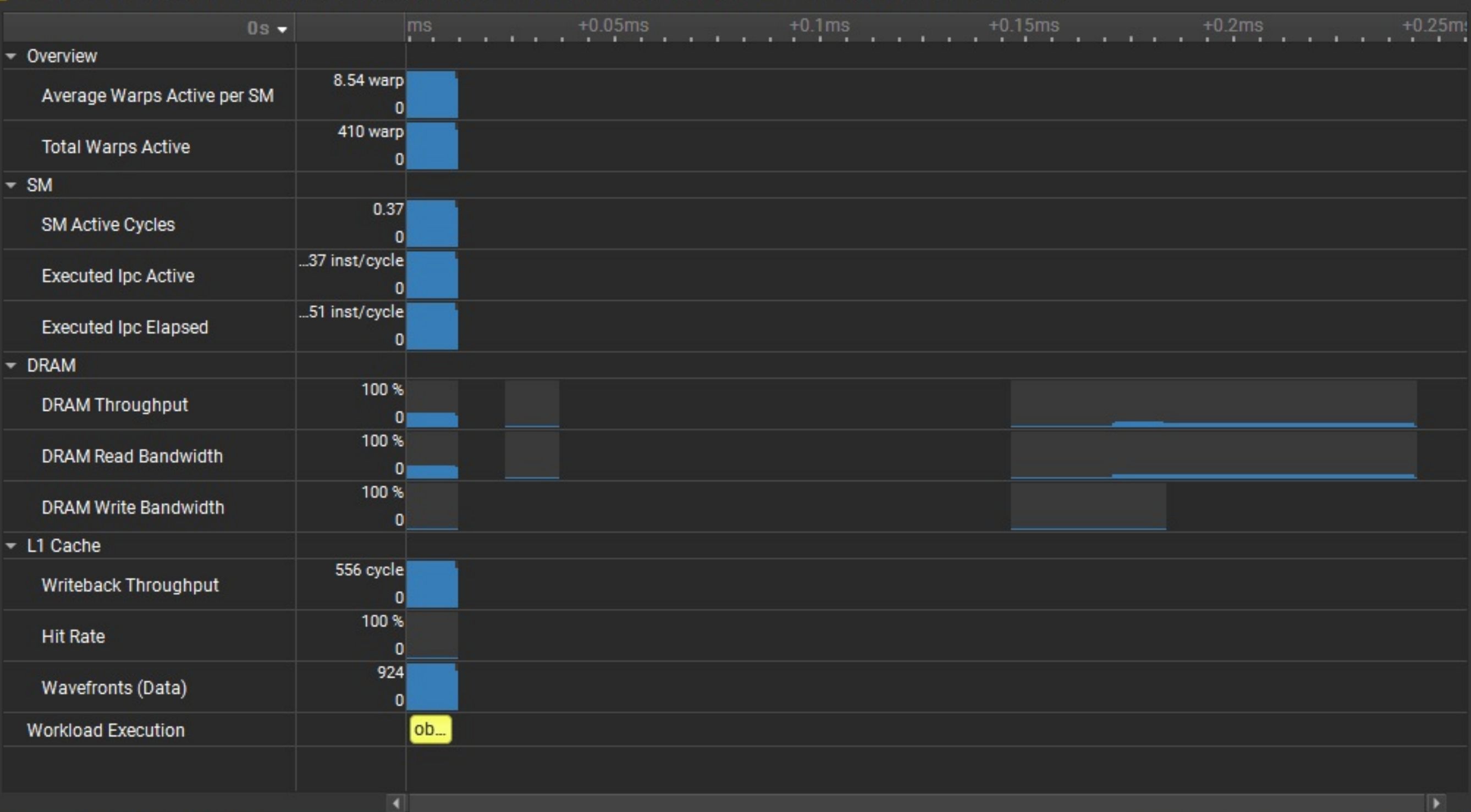


PM Sampling

Timeline view of PM metrics sampled periodically over the workload duration. Data is collected across multiple passes. Use this section to understand how workload behavior changes over its runtime.

Maximum Sampling Interval [cycle]	20,000	# Pass Groups	1
Maximum Buffer Size [Kbyte]	262.14	-	-

PM Sampling Data Sampling interval is 1.2x of the workload duration, which likely results in no or very few collected samples.



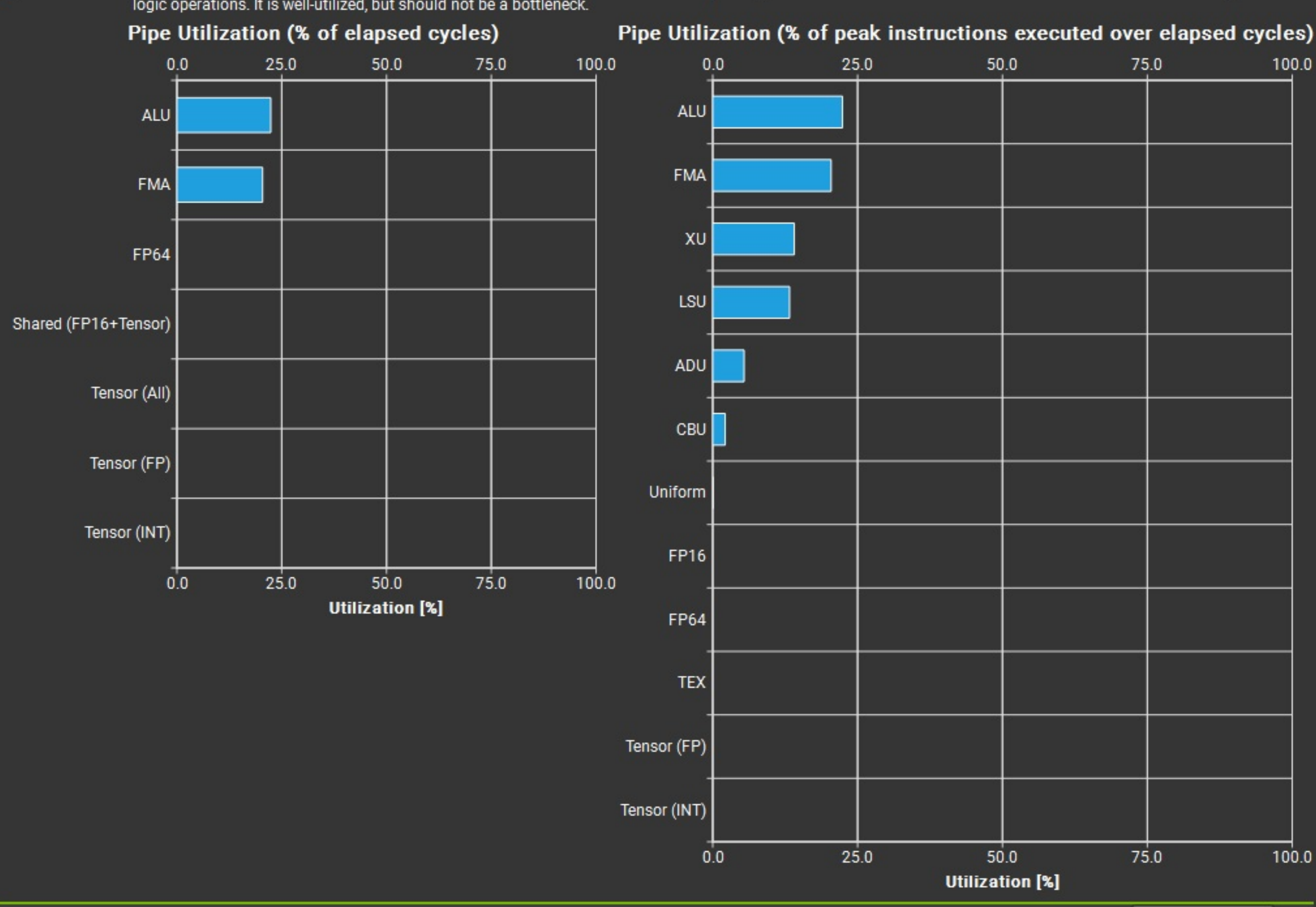
Compute Workload Analysis

Pipe Utilization (Elapsed Cycles)

Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.

Executed Ipc Elapsed [inst/cycle]	1.17	SM Busy [%]	29.66
Executed Ipc Active [inst/cycle]	1.11	Issue Slots Busy [%]	29.66
Issued Ipc Active [inst/cycle]	1.12		

Balanced ALU is the highest-utilized pipeline (22.4%) based on elapsed cycles in the workload, taking into account the rates of its different instructions. It executes integer and logic operations. It is well-utilized, but should not be a bottleneck.



Memory Workload Analysis

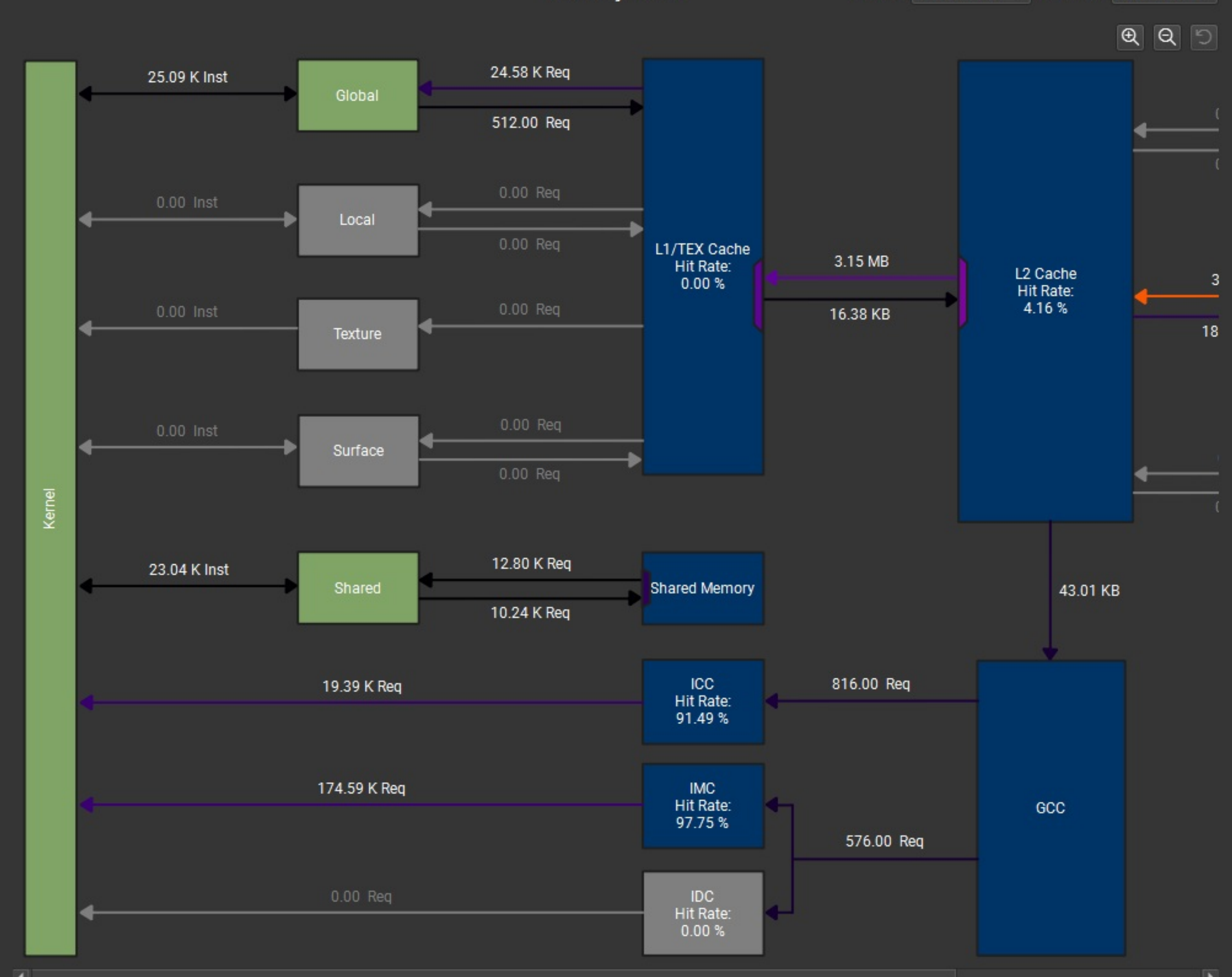
Memory Chart

Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit.

Memory Throughput [Gbyte/s]	318.30	Mem Busy [%]	19.56
L1/TEX Hit Rate [%]	0	Max Bandwidth [%]	65.78
L2 Hit Rate [%]	4.16	Mem Pipes Busy [%]	13.26
Local Memory Spilling Requests	0	Shared Memory Spilling Requests	0
Local Memory Spilling Request Overhead [%]	0	Shared Memory Spilling Request Overhead [%]	0
L2 Sector Promotion Misses [%]	0.00	-	-

Memory Chart

Values: Transfer Size — Inactivity: Greyed Out



Scheduler Statistics

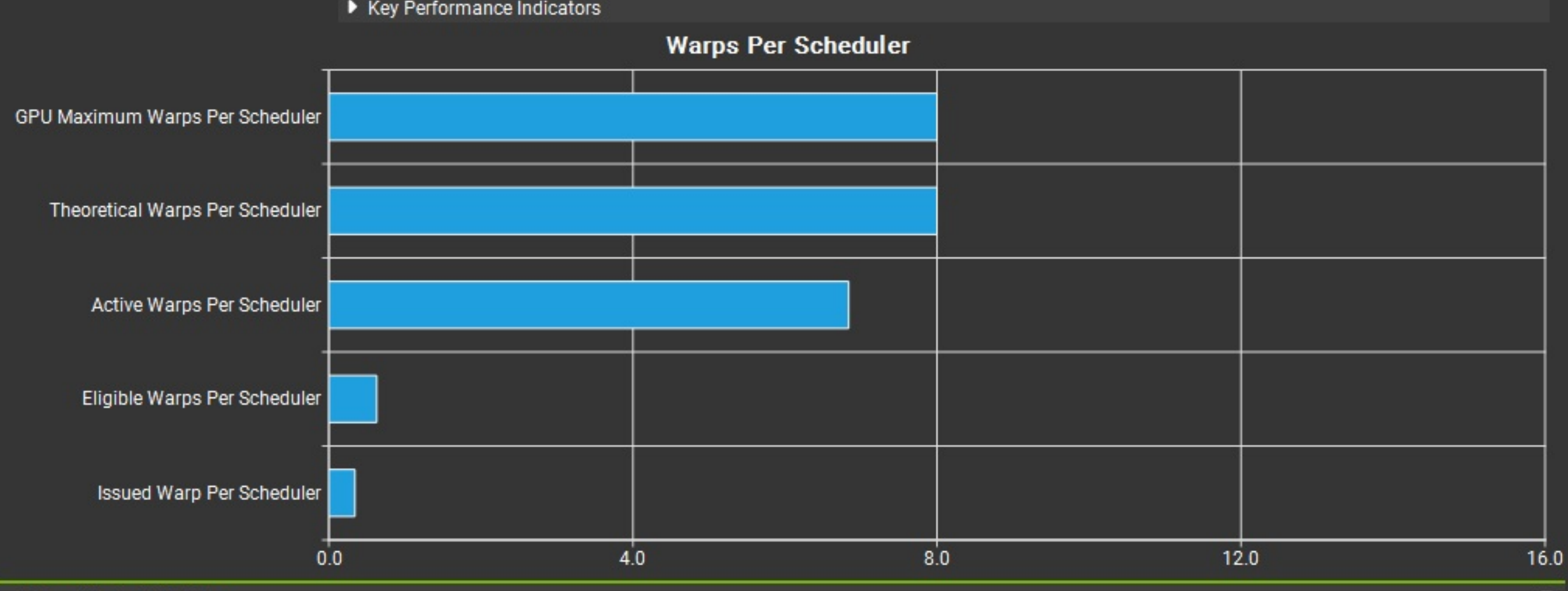
Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.

Active Warps Per Scheduler [warp]	6.84	No Eligible [%]	65.69
Eligible Warps Per Scheduler [warp]	0.63	One or More Eligible [%]	34.31
Issued Warp Per Scheduler	0.34		

Issue Slot Utilization
Est. Local Speedup: 34.22% Every scheduler is capable of issuing one instruction per cycle, but for this workload each scheduler only issues an instruction every 2.9 cycles. This might leave hardware resources underutilized and may lead to less optimal performance. Out of the maximum of 8 warps per scheduler, this workload allocates an average of 6.84 active warps per scheduler, but only an average of 0.63 warps were eligible per cycle. Eligible warps are the subset of active warps that are ready to issue their next instruction. Every cycle with no eligible warp results in no instruction being issued and the issue slot remains unused. To increase the number of eligible warps, avoid possible load imbalances due to highly different execution durations per warp. Reducing stalls indicated on the Warp State Statistics and Source Counters sections can help, too.

Key Performance Indicators

Warps Per Scheduler



Warp State Statistics

Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle. When executing a kernel with mixed library and user code, these metrics show the combined values.

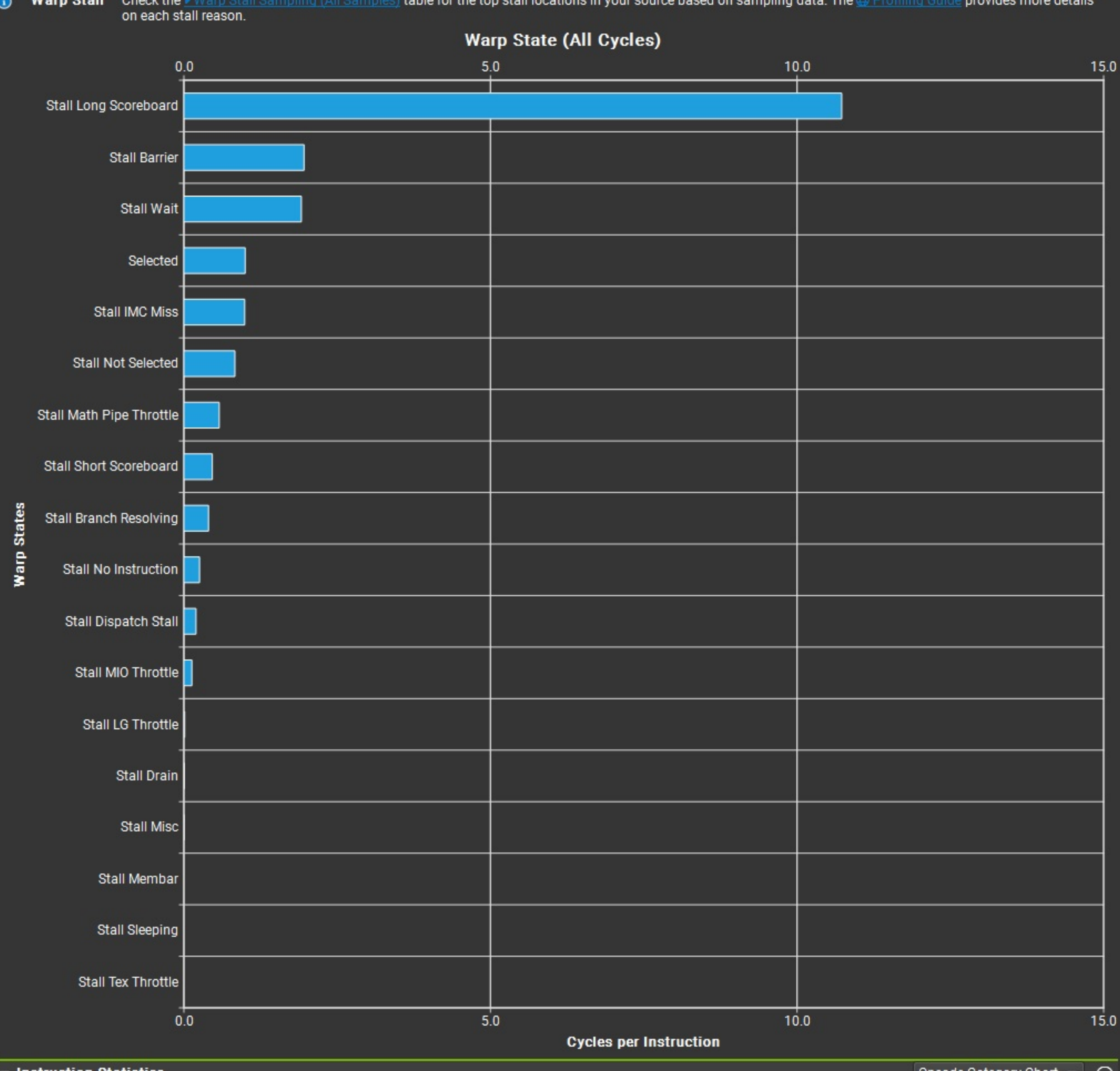
Warp Cycles Per Issued Instruction [cycle]	19.93	Avg. Active Threads Per Warp	31.56
Warp Cycles Per Executed Instruction [cycle]	20.18	Avg. Not Predicated Off Threads Per Warp	29.06

Long Scoreboard Stalls
Est. Speedup: 34.22% On average, each warp of this workload spends 10.7 cycles being stalled waiting for a scoreboard dependency on a L1TEX (local, global, surface, texture) operation. Find the instruction producing the data being waited upon to identify the culprit. To reduce the number of cycles waiting on L1TEX data accesses verify the memory access patterns are optimal for the target architecture, attempt to increase cache hit rates by increasing data locality (coalescing), or by changing the cache configuration. Consider moving frequently used data to shared memory. This stall type represents about 53.8% of the total average of 19.9 cycles between issuing two instructions.

Key Performance Indicators

Warp Stall Check the Warp Stall Sampling (All Samples) table for the top stall locations in your source based on sampling data. The Profiling Guide provides more details on each stall reason.

Warp State (All Cycles)



Instruction Statistics

Opcode Category Chart

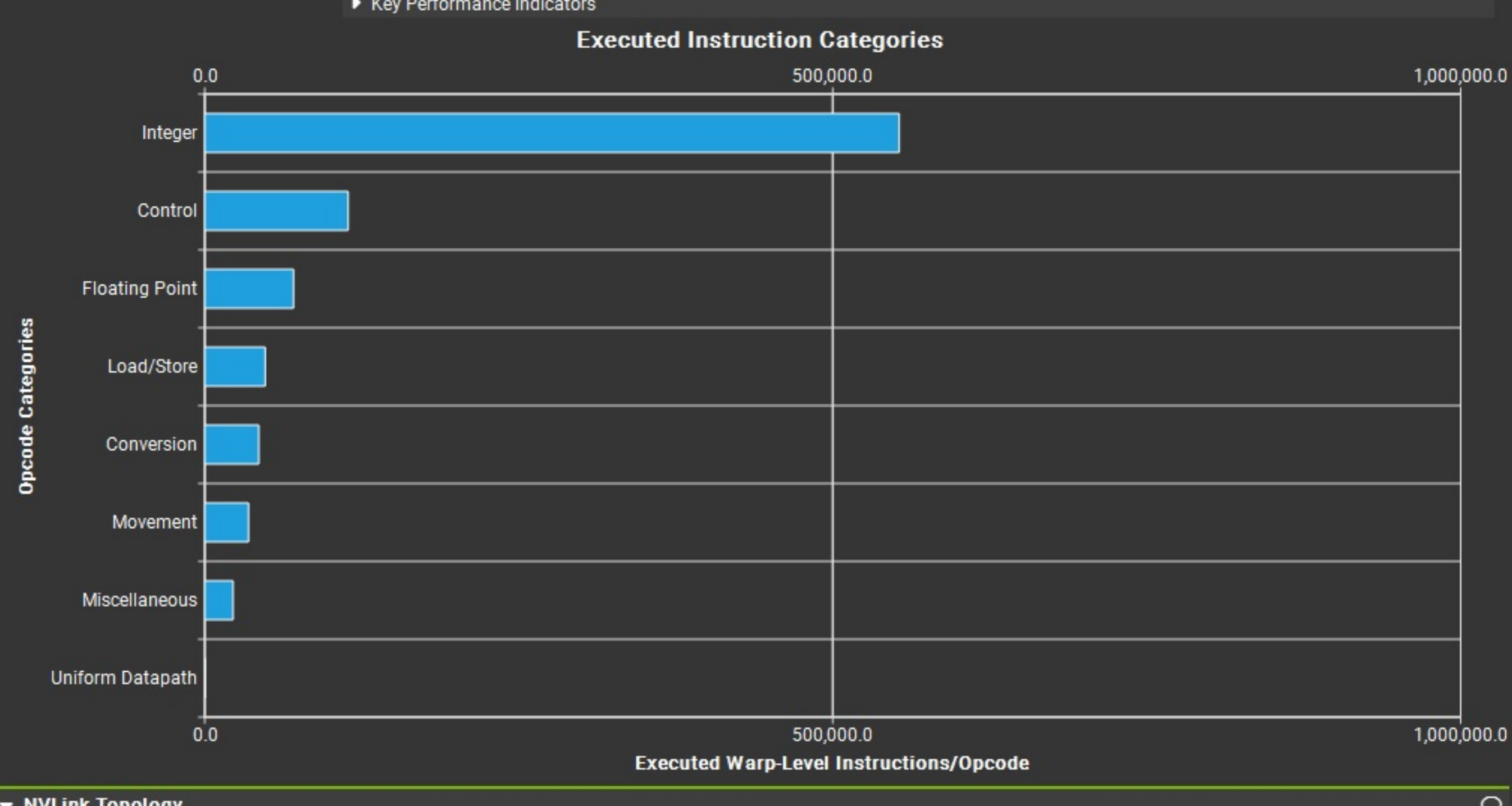
Statistics of the executed low-level assembly instructions (SASS). The instruction mix provides insight into the types and frequency of the executed instructions. A narrow mix of instruction types implies a dependency on few instruction pipelines, while others remain unused. Using multiple pipelines allows hiding latencies and enables parallel execution. Note that 'Instructions/Opcode' and 'Executed Instructions' are measured differently and can diverge if cycles are spent in system calls.

Executed Instructions [inst]	886,784	Avg. Executed Instructions Per Scheduler [inst]	4,618.67
Issued Instructions [inst]	897,813	Avg. Issued Instructions Per Scheduler [inst]	4,676.11
Local Memory Spilling Requests [inst]	0	Shared Memory Spilling Requests [inst]	0

FP32 Non-Fused Instructions
Est. Speedup: 5.37% This workload executes 28672 fused and 31744 non-fused FP32 instructions. By converting pairs of non-fused instructions to their fused, higher-throughput equivalent, the achieved FP32 performance could be increased by up to 26% (relative to its current performance).

Key Performance Indicators

Executed Instruction Categories



NVLink Topology

NVLink Topology diagram shows logical NVLink connections with transmit/receive throughput.

NVLink Topology

The system does not have any NVLink connections.

NVLink Tables

Detailed tables with properties for each NVLink.

Logical NVLink Properties

The system does not have any NVLink connections.

NUMA Affinity

Non-uniform memory access (NUMA) affinities based on compute and memory distances for all GPUs.

NUMA ID Table

NUMA information is not available on the target system.

Launch Statistics

Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.

Grid Size	256	Function Cache Configuration	CachePreferNone
Registers Per Thread [register/thread]	40	Static Shared Memory Per Block [Kbyte/block]	2.05
Block Size	256	Dynamic Shared Memory Per Block [byte/block]	0
Threads [thread]	65,536	Driver Shared Memory Per Block [byte/block]	0
Waves Per SM	1.33	Shared Memory Configuration Size [Kbyte]	32.77
Uses Green Context	0	Stack Size	1,024
# SMs [SM]	48	# TPCs	24
Enabled TPC IDs	all	-	-

Tail Effect
Est. Speedup: 50.00% A wave of thread blocks is defined as the maximum number of blocks that can be executed in parallel on the target GPU. The number of blocks in a wave depends on the number of multiprocessors and the theoretical occupancy of the kernel. This kernel launch results in 1 full waves and a partial wave of 64 thread blocks. Under the assumption of a uniform execution duration of all thread blocks, this partial wave may account for up to 50.0% of the total runtime of this kernel. Try launching a grid with no partial wave. The overall impact of this tail effect also lessens with the number of full waves executed for a grid. See the Hardware Model description for more details on launch configurations.

Key Performance Indicators

Metric Name	Value	Guidance
launch__waves_per_multiprocessor	1.33333	Decrease the number of partial waves (the fractional par...

Occupancy

% Occupancy Graphs

Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.

Theoretical Occupancy [%]	100	Block Limit Registers [block]	6
Theoretical Active Warps per SM [warp]	32	Block Limit Shared Mem [block]	16
Achieved Occupancy [%]	68.25	Block Limit Warps [block]	4
Achieved Active Warps Per SM [warp]	21.84	Block Limit SM [block]	16

Achieved Occupancy
Est. Speedup: 31.75% The difference between calculated theoretical (100.0%) and measured achieved occupancy (68.3%) can be the result of warp scheduling overheads or workload imbalances during the kernel execution. Load imbalances can occur between warps within a block as well as across blocks of the same kernel. See the CUDA Best Practices Guide for more details on optimizing occupancy.

Key Performance Indicators

GPU and Memory Workload Distribution

Analysis of workload distribution in active cycles of SM, SMP, SMSP, L1 & L2 caches, and DRAM

Average SM Active Cycles [cycle]	16,705.04	Average L1 Active Cycles [cycle]	16,705.04
Average L2 Active Cycles [cycle]	11,051.66	Average SMSP Active Cycles [cycle]	13,628.23
Average DRAM Active Cycles [cycle]	52,201	Total SM Elapsed Cycles [cycle]	756,784
Total L1 Elapsed Cycles [cycle]	756,784	Total L2 Elapsed Cycles [cycle]	517,888
Total SMSP Elapsed Cycles [cycle]	3,027,136	Total DRAM Elapsed Cycles [cycle]	634,880

L2 Slices Workload Imbalance
Est. Speedup: 7.05% One or more L2 Slices have a much higher number of active cycles than the average number of active cycles. Maximum instance value is 10.32% above the average, while the minimum instance value is 2.58% below the average.

Key Performance Indicators

Metric Name	Value	Guidance
lts__cycles_active.avg	11051.7	Balancing the number of active cycles across L2 Slic...

Workload Distribution

	Average	Min	Max	Sum
SM Active Cycles	16,705.04	15,579	17,421	801,842
SMSP Active Cycles	13,628.23	12,749	14,311	2,616,621
L1 Active Cycles	16,705.04	15,579	17,421	801,842
L2 Active Cycles	11,051.66	10,766	12,324	353,653
DRAM Active Cycles	52,201	51,836	52,548	417,608

Source Counters

Source metrics, including branch efficiency and sampled warp stall reasons. Warp Stall Sampling metrics are periodically sampled over the kernel runtime. They indicate when warps were stalled and couldn't be scheduled. See the documentation for a description of all stall reasons. Only focus on stalls if the schedulers fail to issue every cycle.

Branch Instructions [inst]	73,216	Branch Efficiency [%]	97.40
Branch Instructions Ratio [%]	0.08	Avg. Divergent Branches [branches]	6.67

Warp Stall Sampling (Not-issued Samples)

Location	Value	Value (%)
0xb00e5bda0 in ob1_re...	132	66
0xb00e5c4d0 in ob1_re...	21	11
0xb00e5c4b0 in ob1_re...	6	3
0xb00e5c4e0 in ob1_re...	5	3
0xb00e5c400 in ob1_re...	5	3

Most Instructions Executed

Location	Value	Value (%)
0xb00e5c4a0 in ob1_re...	17,664	2
0xb00e5c4e0 in ob1_re...	16,384	2
0xb00e5c4d0 in ob1_re...	16,384	2
0xb00e5c4c0 in ob1_re...	16,384	2
0xb00e5c4b0 in ob1_re...	16,384	2

Follow the *rules outputs* to get guidance on how to navigate through the report and quickly discover performance bottlenecks in this kernel.
You could also disable individual sections to focus on selected performance aspects and make profiling faster.