	Report	Result	Size	Time	Cycles	GPU	SM Frequency	Process	Attributes
Current	jacobi2D_shared_mem_1024_profiling	572 - jacobi_shared_kernel	(64, 64, 1)x(16, 16, 1)	27.10 us	63,792	0 - NVIDIA GeForce RTX 5060 Ti	2.34 Ghz	[350158] jacobi2D_shared_mem_1024.exe	
Baseline 1	jacobi2D_pointer_swap_1024_profiling	572 - jacobi_kernel	(32, 128, 1)x(32, 8, 1)	14.59 us	33,534	0 - NVIDIA GeForce RTX 5060 Ti	2.28 Ghz	[345743] jacobi2D_pointer_swap_1024.exe	

Summary | Details | Source | Context | Comments | Raw | Session

Compare | Tools | View | Export

GPU Speed Of Light Throughput

GPU Throughput Chart

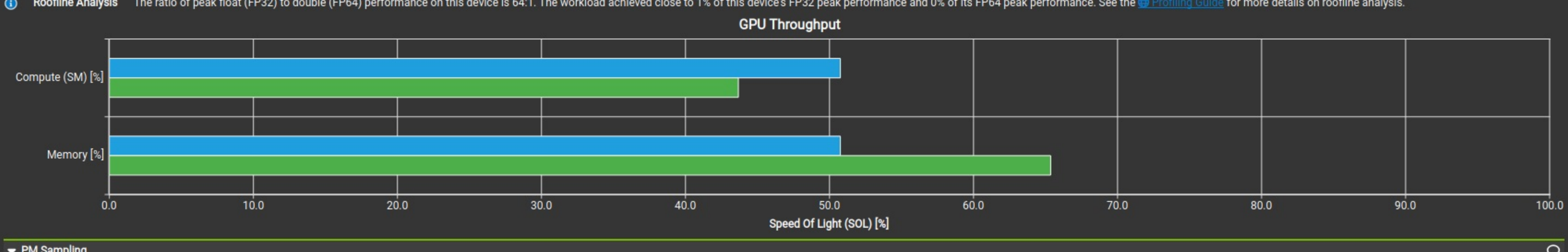
High-level overview of the throughput for compute and memory resources of the GPU. For each unit, the throughput reports the achieved percentage of utilization with respect to the theoretical maximum. Breakdowns show the throughput for each individual sub-metric of Compute and Memory to clearly identify 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 [%]	50.76 (+16.22%)	Duration [us]	27.10 (+85.75%)
Memory Throughput [%]	50.76 (-22.34%)	Elapsed Cycles [cycle]	63,792 (+90.23%)
L1/TEX Cache Throughput [%]	53.54 (+10.93%)	SM Active Cycles [cycle]	60,114.81 (+99.52%)
L2 Cache Throughput [%]	36.15 (-21.17%)	SM Frequency [Ghz]	2.34 (+2.52%)
DRAM Throughput [%]	35.15 (-46.22%)	DRAM Frequency [Ghz]	13.77 (+0.12%)

Latency Issue This workload exhibits low compute throughput and memory bandwidth utilization relative to the peak performance of this device. Achieved compute throughput and/or memory bandwidth below 60.0% of peak typically indicate latency issues. Look at Scheduler Statistics and Warp State Statistics for potential reasons.

Key Performance Indicators

Roofline Analysis The ratio of peak float (FP32) to double (FP64) performance on this device is 64:1. The workload achieved close to 1% of this device's FP32 peak performance and 0% of its FP64 peak performance. See the Profiling Guide for more details on roofline analysis.

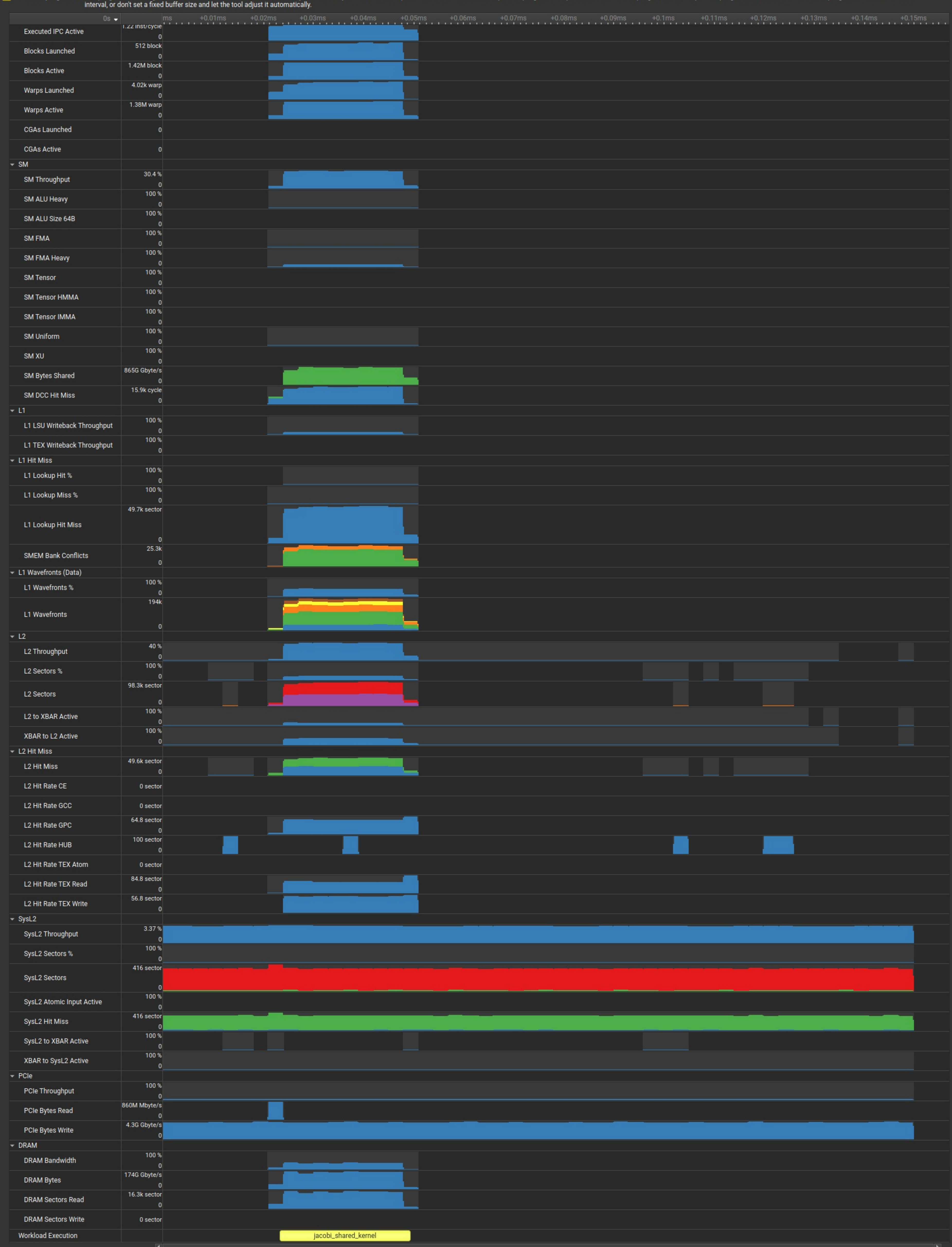


PM Sampling

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 [us]	3 (+0.00%)	# Pass Groups	1 (+0.00%)
Maximum Buffer Size [Mbyte]	4.13 (-50.00%)	-	-

PM Sampling Data Sampling interval is larger than 10% of the workload duration, which likely results in very few collected samples. For better results, use the --pm-sampling-interval option to reduce the sampling interval. Use --pm-sampling-buffer-size to increase the sampling buffer size for the smaller interval, or don't set a fixed buffer size and let the tool adjust it automatically.



Compute Workload Analysis

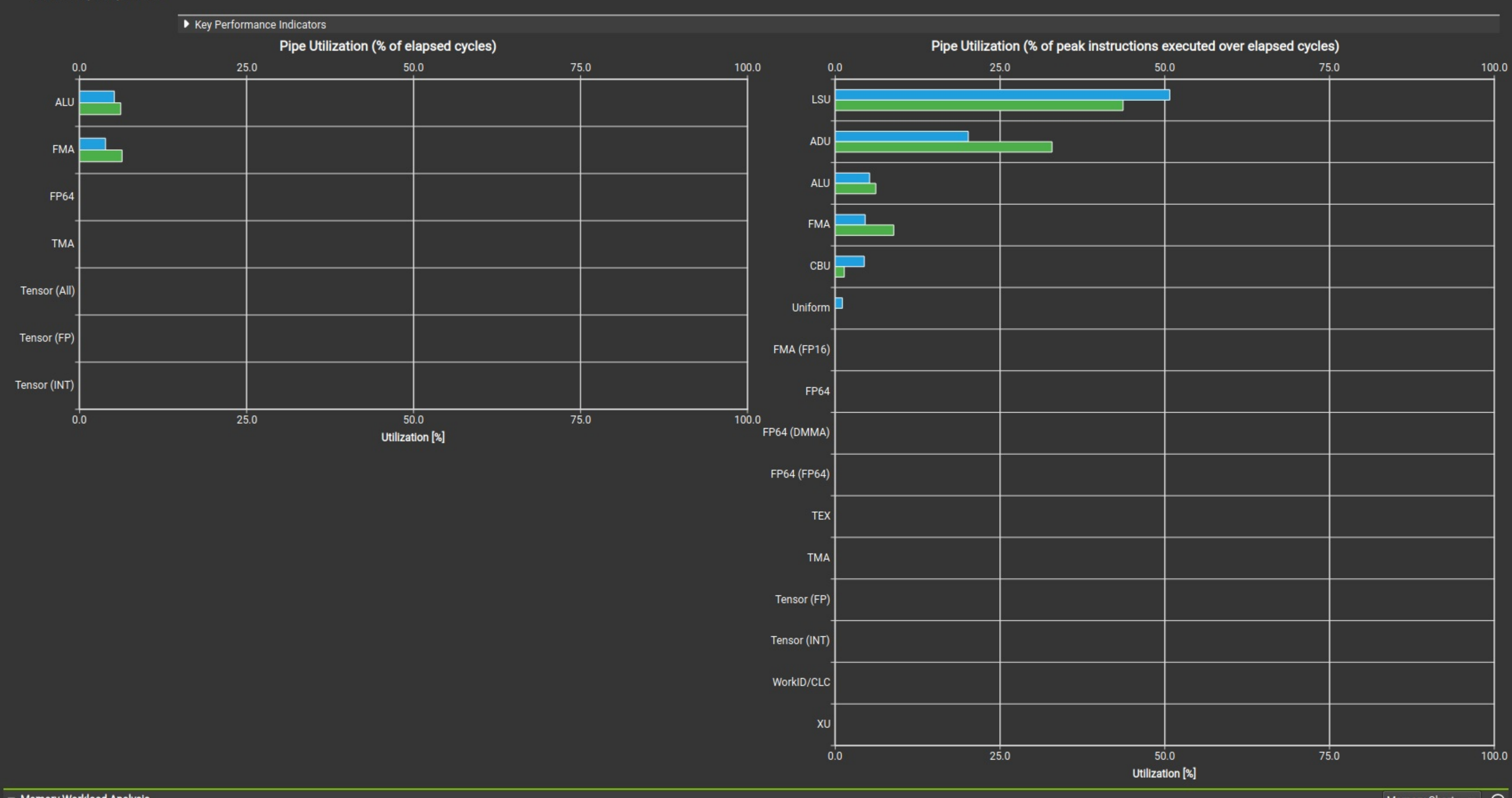
Pipe Utilization (Elapsed Cycles)

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]	1.11 (-0.89%)	SM Busy [%]	27.74 (-0.89%)
Executed Ipc Active [inst/cycle]	1.17 (-5.40%)	Issue Slots Busy [%]	27.74 (-0.89%)
Issued Ipc Active [inst/cycle]	1.17 (-5.40%)		

Low Utilization
Est. Local Speedup: 94.80% All compute pipelines are under-utilized. Either this workload is very small or it doesn't issue enough warps per scheduler. Check the Launch Statistics and Scheduler Statistics sections for further details.

Key Performance Indicators



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]	154.90 (-46.15%)	Mem Busy [%]	41.15 (+10.17%)
L1/TEX Hit Rate [%]	0.40 (-99.31%)	Max Bandwidth [%]	50.76 (-22.34%)
L2 Hit Rate [%]	51.79 (+93.79%)	Mem Pipes Busy [%]	50.76 (+16.22%)
L2 Compression Input Sectors [sector]	0 (+0.00%)	Local Memory Spilling Requests	0 (+0.00%)
L2 Compression Ratio	0 (+0.00%)	Local Memory Spilling Request Overhead [%]	0 (+0.00%)
L2 Compression Success Rate [%]	0 (+0.00%)	L2 Persisting Size [Mbyte]	6.29 (+0.00%)

L1TEX Global Load Access Pattern
Est. Speedup: 20.74% The memory access pattern for global loads from L1TEX might not be optimal. On average, only 18.9 of the 32 bytes transmitted per sector are utilized by each thread. This could possibly be caused by a stride between threads. Check the Source Counters section for uncoalesced global loads.

Key Performance Indicators

L1TEX Global Store Access Pattern
Est. Speedup: 0.10% The memory access pattern for global stores to L1TEX might not be optimal. On average, only 31.9 of the 32 bytes transmitted per sector are utilized by each thread. This could possibly be caused by a stride between threads. Check the Source Counters section for uncoalesced global stores.

Key Performance Indicators

Shared Load Bank Conflicts
Est. Speedup: 27.13% The memory access pattern for shared loads might not be optimal and causes on average a 2.0 - way bank conflict across all 163840 shared load requests.This results in 168290 bank conflicts, which represent 50.67% of the overall 332125 wavefronts for shared loads. Check the Source Counters section for uncoalesced shared loads.

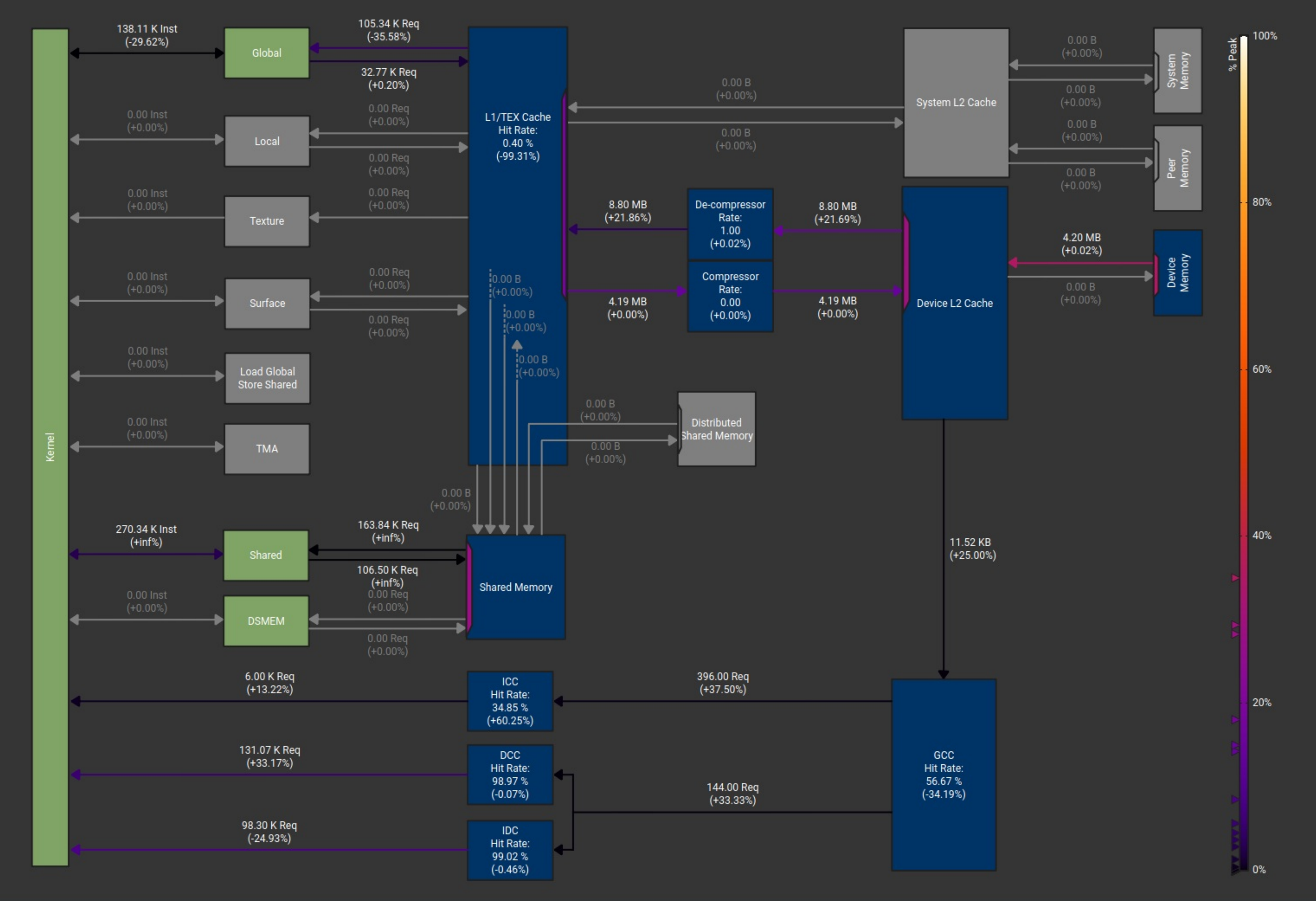
Key Performance Indicators

Shared Store Bank Conflicts
Est. Speedup: 13.36% The memory access pattern for shared stores might not be optimal and causes on average a 1.3 - way bank conflict across all 106496 shared store requests.This results in 35381 bank conflicts, which represent 24.96% of the overall 141770 wavefronts for shared stores. Check the Source Counters section for uncoalesced shared stores.

Key Performance Indicators

Memory Chart

Values: Transfer Size Inactivity: Greyed Out



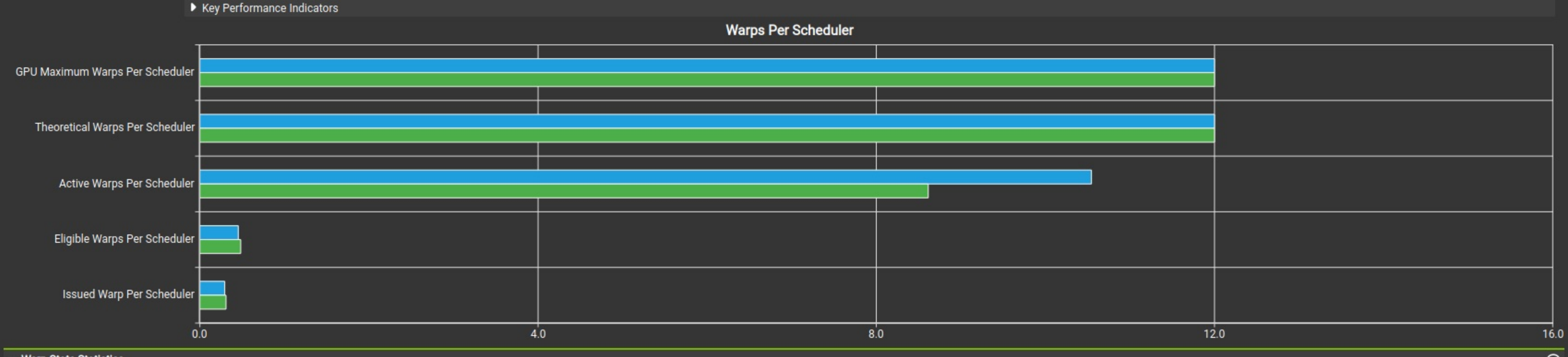
Scheduler Statistics

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]	10.54 (+22.41%)	No Eligible [%]	70.59 (+2.18%)
Eligible Warps Per Scheduler [warp]	0.45 (-6.00%)	One or More Eligible [%]	29.41 (-4.87%)
Issued Warp Per Scheduler	0.29 (-4.87%)		

Issue Slot Utilization
Est. Local Speedup: 49.24% Every scheduler is capable of issuing one instruction per cycle, but for this workload each scheduler only issues an instruction every 3.4 cycles. This might leave hardware resources underutilized and may lead to less optimal performance. Out of the maximum of 12 warps per scheduler, this workload allocates an average of 10.54 active warps per scheduler, but only an average of 0.45 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



Warp State Statistics

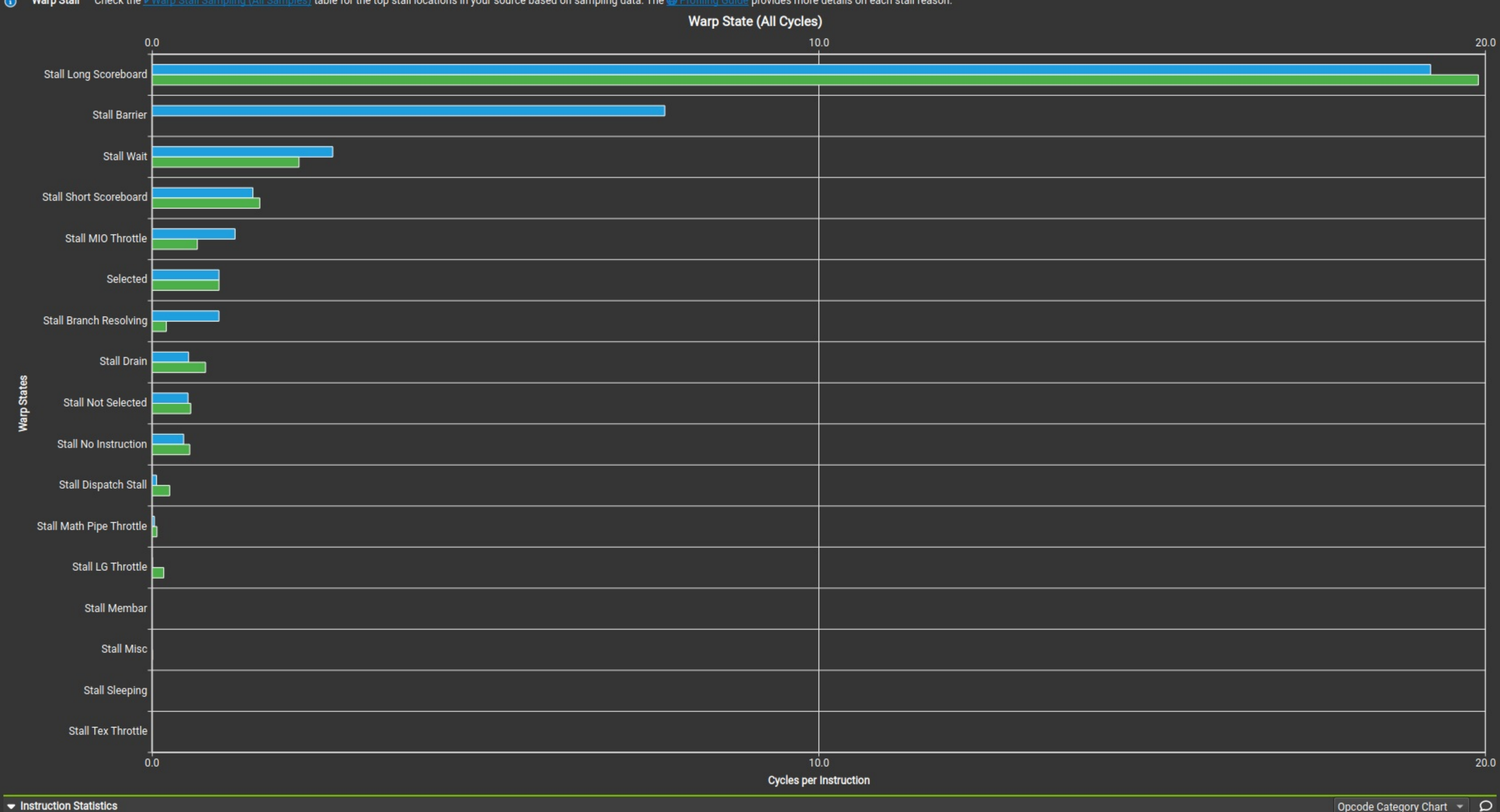
Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism 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]	35.85 (+28.68%)	Avg. Active Threads Per Warp	26.75 (-16.30%)
Warp Cycles Per Executed Instruction [cycle]	35.85 (+28.68%)	Avg. Not Predicated Off Threads Per Warp	25.62 (-17.84%)

Long Scoreboard Stalls
Est. Speedup: 49.24% On average, each warp of this workload spends 19.2 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 53.5% of the total average of 35.8 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.



Instruction Statistics

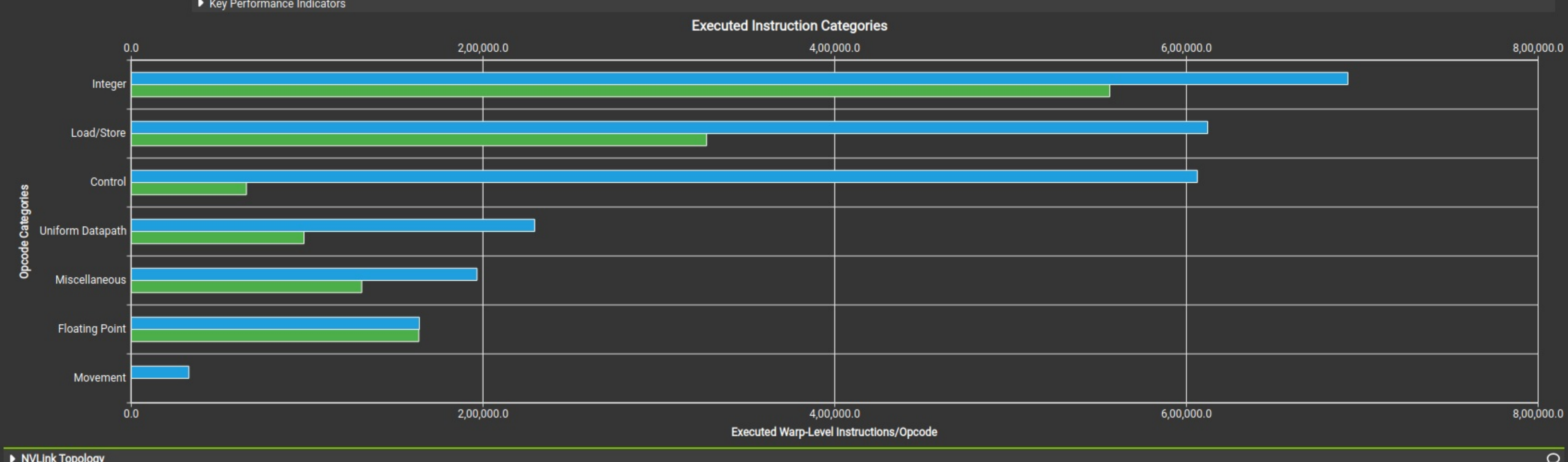
Opcode Category Chart

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]	25,32,736 (+88.74%)	Avg. Executed Instructions Per Scheduler [inst]	17,588.44 (+88.74%)
Issued Instructions [inst]	25,32,736 (+88.74%)	Avg. Issued Instructions Per Scheduler [inst]	17,588.44 (+88.74%)

FP32 Non-Fused Instructions
Est. Speedup: 1.94% This kernel executes 0 fused and 163840 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 50% (relative to its current performance).

Key Performance Indicators



NVLink Topology

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

NVLink Tables

Detailed tables with properties for each NVLink.

NUMA Affinity

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

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	4,096 (+0.00%)	Function Cache Configuration	CachePreferNone (CachePreferNone)
Cluster Size	0 (+0.00%)	Preferred Cluster Size	0 (+0.00%)
Registers Per Thread [register/thread]	18 (-10.00%)	Cluster Scheduling Policy	PolicySpread (PolicySpread)
Static Shared Memory Per Block [Kbyte/block]	1.30 (+inf%)	Block Size	256 (+0.00%)
Dynamic Shared Memory Per Block [byte/block]	0 (+0.00%)	Threads [thread]	10,48,576 (+0.00%)
Driver Shared Memory Per Block [Kbyte/block]	1.02 (+0.00%)	Waves Per SM	18.96 (+0.00%)
Shared Memory Configuration Size [Kbyte]	32.77 (+100.00%)	Uses Green Context	0 (+0.00%)
Stack Size	1,024 (+0.00%)	# SMs [SM]	36 (+0.00%)
# TPCs	18 (+0.00%)	Enabled TPC IDs	all (all)

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 (+0.00%)	Block Limit Registers [block]	10 (+0.00%)
Theoretical Active Warps per SM [warp]	48 (+0.00%)	Block Limit Shared Mem [block]	13 (-18.75%)
Achieved Occupancy [%]	89.01 (+20.94%)	Block Limit Warps [block]	6 (+0.00%)
Achieved Active Warps Per SM [warp]	42.73 (+20.94%)	Block Limit SM [block]	24 (+0.00%)
Cluster Occupancy [%]	0 (+0.00%)	Block Limit Barriers [block]	24 (+0.00%)
Max Active Clusters [cluster]	0 (+0.00%)	Max Cluster Size [block]	8 (+0.00%)
Overall GPU Occupancy [%]	0 (+0.00%)		

Achieved Occupancy
Est. Speedup: 10.99% The difference between calculated theoretical (100.0%) and measured achieved occupancy (89.0%) 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]	60,114.81 (+99.52%)	Average L1 Active Cycles [cycle]	60,114.81 (+99.52%)
Average L2 Active Cycles [cycle]	53,770.62 (+103.43%)	Average SMSP Active Cycles [cycle]	59,795.12 (+98.41%)
Average DRAM Active Cycles [cycle]	1,31,200 (+0.02%)	Total SM Elapsed Cycles [cycle]	22,82,820 (+90.43%)
Total L1 Elapsed Cycles [cycle]	22,82,820 (+90.43%)	Total L2 Elapsed Cycles [cycle]	9,09,024 (+89.82%)
Total SMSP Elapsed Cycles [cycle]	91,31,280 (+90.43%)	Total DRAM Elapsed Cycles [cycle]	14,92,992 (+85.97%)

Workload Distribution

	Average	Min	Max	Sum
SM Active Cycles	60,114.81 (+99.52%)	59,564 (+100.10%)	60,778 (+99.18%)	21,64,133 (+99.52%)
SMSP Active Cycles	59,795.12 (+98.41%)	59,179 (+100.67%)	60,418 (+98.22%)	86,10,498 (+98.41%)
L1 Active Cycles	60,114.81 (+99.52%)	59,564 (+100.10%)	60,778 (+99.18%)	21,64,133 (+99.52%)
L2 Active Cycles	53,770.62 (+103.43%)	53,350 (+104.43%)	54,523 (+103.26%)	8,60,330 (+103.43%)
DRAM Active Cycles	1,31,200 (+0.02%)	1,31,072 (+0.00%)	1,31,328 (+0.07%)	5,24,800 (+0.02%)

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]	4,75,136 (+625.71%)	Branch Efficiency [%]	64 (+inf%)
Branch Instructions Ratio [%]	0.19 (+284.49%)	Avg. Divergent Branches [branches]	512 (+inf%)

Uncoalesced Global Accesses
Est. Speedup: 15.00% This kernel has uncoalesced global accesses resulting in a total of 64512 excessive sectors (16% of the total 407040 sectors). Check the L2 Theoretical Sectors Global Excessive table for the primary source locations. The CUDA Programming Guide has additional information on reducing uncoalesced device memory accesses.

Key Performance Indicators

L2 Theoretical Sectors Global Excessive

Location	Value	Value (%)
0x716b3f767970 in jacobi_shared_kernel	32,256	50
0x716b3f7678b0 in jacobi_shared_kernel	32,256	50
0x716b3f767a80 in jacobi_shared_kernel	0	0
0x716b3f767830 in jacobi_shared_kernel	0	0
0x716b3f767730 in jacobi_shared_kernel	0	0

Uncoalesced Shared Accesses
Est. Speedup: 39.84% This kernel has uncoalesced shared accesses resulting in a total of 195968 excessive wavefronts (42% of the total 466304 wavefronts). Check the L1 Wavefronts Shared Excessive table for the primary source locations. The CUDA Best Practices Guide has an example on optimizing shared memory accesses.

Key Performance Indicators

L1 Wavefronts Shared Excessive

Location	Value	Value (%)
0x716b3f7676a0 in jacobi_shared_kernel	32,768	17
0x716b3f767a10 in jacobi_shared_kernel	32,640	17
0x716b3f767a00 in jacobi_shared_kernel	32,640	17
0x716b3f7679f0 in jacobi_shared_kernel	32,640	17
0x716b3f7679e0 in jacobi_shared_kernel	32,640	17

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.