Page: Details | Result: 1 - 129 - matrixVectorMulKernel | Add Baseline | Apply Rules | Occupancy Calculator | Source Comparison | Save as PDF

	Result	Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current	129 - matrixVectorMulKernel (6250, 8, 1)x(16, 16, 1)	171.46 msecond	26,74,61,656	37	0 - NVIDIA GeForce RTX 3090 Ti	1.56 cycle/nsecond	8.6	[6025] Decrypt_Serial

GPU Speed Of Light Throughput

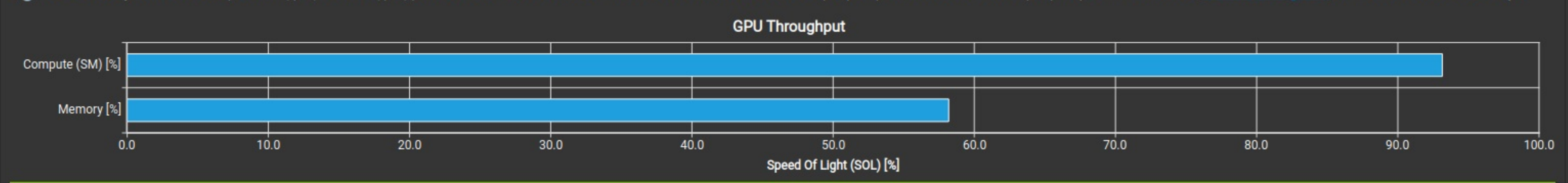
GPU Throughput Chart

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 [%]	93.15	Duration [msecond]	171.46
Memory Throughput [%]	58.19	Elapsed Cycles [cycle]	26,74,61,656
L1/TEX Cache Throughput [%]	58.26	SM Active Cycles [cycle]	26,71,43,213.19
L2 Cache Throughput [%]	3.24	SM Frequency [cycle/nsecond]	1.56
DRAM Throughput [%]	3.37	DRAM Frequency [cycle/nsecond]	10.24

High Throughput The kernel is utilizing greater than 80.0% of the available compute or memory performance of the device. To further improve performance, work will likely need to be shifted from the most utilized to another unit. Start by analyzing workloads in the Compute Workload Analysis section.

Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 64:1. The kernel achieved 0% of this device's fp32 peak performance and 0% of its fp64 peak performance. See the Kernel 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 workload behavior changes over its runtime.

Compute Workload Analysis

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]	2.92	SM Busy [%]	93.26
Executed Ipc Active [inst/cycle]	2.92	Issue Slots Busy [%]	73.07
Issued Ipc Active [inst/cycle]	2.92		

Very High Utilization ALU is the highest-utilized pipeline (93.3%) based on active cycles, taking into account the rates of its different instructions. It executes integer and logic operations. The pipeline is over-utilized and likely a performance bottleneck. Based on the number of executed instructions, the highest utilized pipeline (93.3%) is ALU. It executes integer and logic operations. Comparing the two, the overall pipeline utilization appears to be caused by frequent, low-latency instructions. See the Kernel Profiling Guide or hover over the pipeline name to understand the workloads handled by each pipeline. The Instruction Statistics section shows the mix of executed instructions in this kernel. Check the Warp State Statistics section for which reasons cause warps to stall.

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/second]	33.15	Mem Busy [%]	58.19
L1/TEX Hit Rate [%]	98.57	Max Bandwidth [%]	24.85
L2 Hit Rate [%]	50.85	Mem Pipes Busy [%]	24.85
L2 Compression Success Rate [%]	0	L2 Compression Ratio	0

L1TEX Global Load Access Pattern Est. Speedup: 5.04% The memory access pattern for global loads in L1TEX might not be optimal. On average, this kernel accesses 2.0 bytes per thread per memory request; but the address pattern, possibly caused by the stride between threads, results in 8.5 sectors per request, or 8.5*32 = 271.8 bytes of cache data transfers per request. The optimal thread address pattern for 2.0 byte accesses would result in 2.0*32 = 64.0 bytes of cache data transfers per request, to maximize L1TEX cache performance. Check the Source Counters section for uncoalesced global loads.

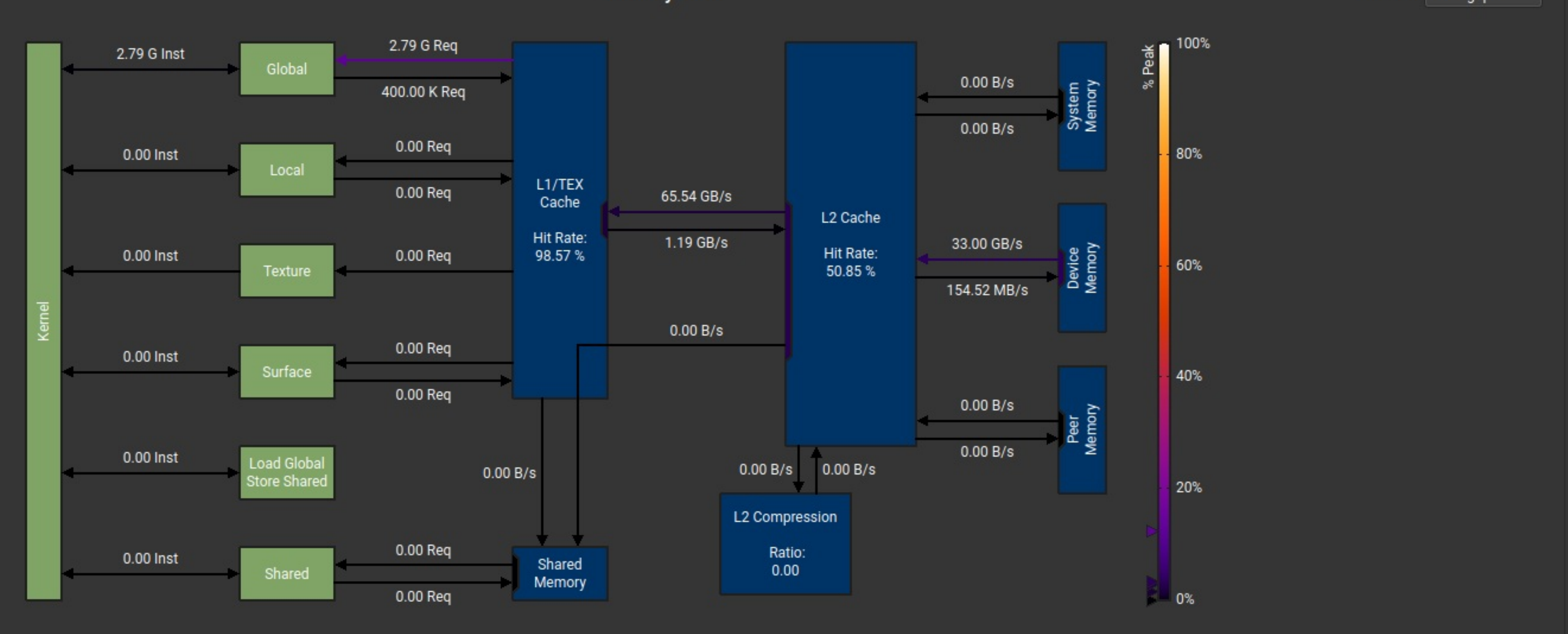
L1TEX Global Store Access Pattern Est. Speedup: 5.77% The memory access pattern for global stores in L1TEX might not be optimal. On average, this kernel accesses 2.0 bytes per thread per memory request; but the address pattern, possibly caused by the stride between threads, results in 16.0 sectors per request, or 16.0*32 = 512.0 bytes of cache data transfers per request. The optimal thread address pattern for 2.0 byte accesses would result in 2.0*32 = 64.0 bytes of cache data transfers per request, to maximize L1TEX cache performance. Check the Source Counters section for uncoalesced global stores.

L2 Load Access Pattern Est. Speedup: 2.11% The memory access pattern for loads from L1TEX to L2 is not optimal. The granularity of an L1TEX request to L2 is a 128 byte cache line. That is 4 consecutive 32-byte sectors per L2 request. However, this kernel only accesses an average of 1.0 sectors out of the possible 4 sectors per cache line. Check the Source Counters section for uncoalesced loads and try to minimize how many cache lines need to be accessed per memory request.

L2 Store Access Pattern Est. Speedup: 0.04% The memory access pattern for stores from L1TEX to L2 is not optimal. The granularity of an L1TEX request to L2 is a 128 byte cache line. That is 4 consecutive 32-byte sectors per L2 request. However, this kernel only accesses an average of 1.0 sectors out of the possible 4 sectors per cache line. Check the Source Counters section for uncoalesced stores and try to minimize how many cache lines need to be accessed per memory request.

Memory Chart

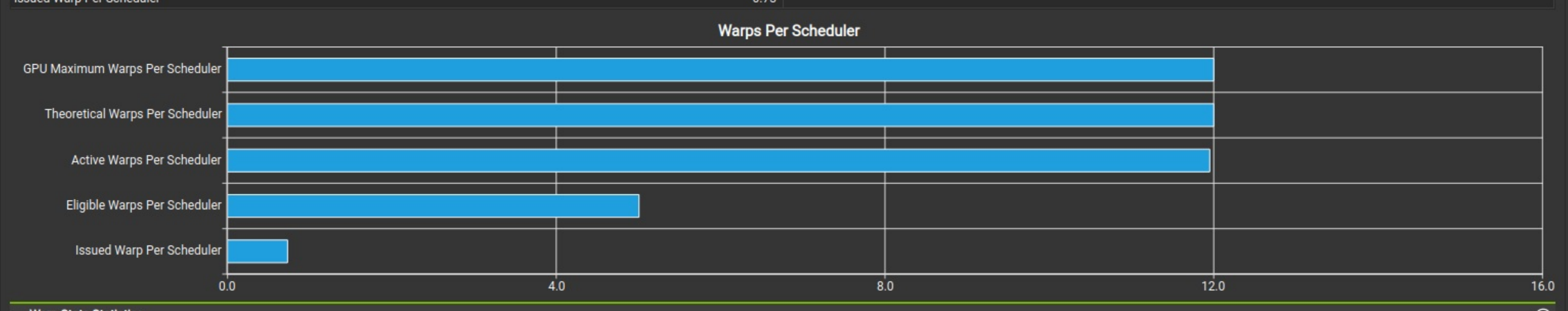
Show As: Throughput



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]	11.95	No Eligible [%]	26.93
Eligible Warps Per Scheduler [warp]	5.01	One or More Eligible [%]	73.07
Issued Warp Per Scheduler	0.73		



Warp State Statistics

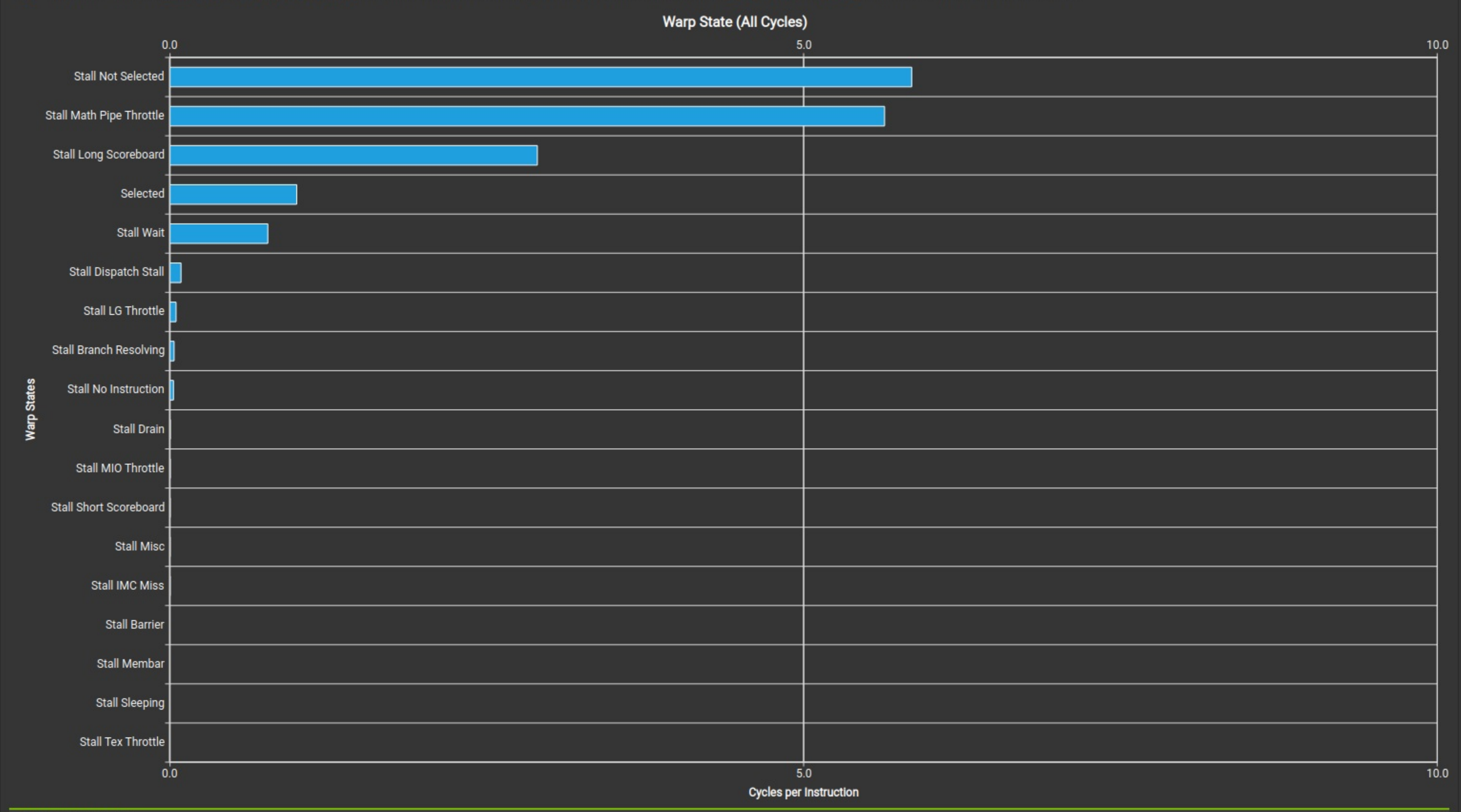
Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism 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]	16.36	Avg. Active Threads Per Warp	32
Warp Cycles Per Executed Instruction [cycle]	16.36	Avg. Not Predicated Off Threads Per Warp	32.00

Not Selected Stalls Est. Local Speedup: 35.78% On average, each warp of this kernel spends 5.9 cycles being stalled waiting for the micro scheduler to select the warp to issue. Not selected warps are eligible warps that were not picked by the scheduler to issue that cycle as another warp was selected. A high number of not selected warps typically means you have sufficient warps to cover warp latencies and you may consider reducing the number of active warps to possibly increase cache coherence and data locality. This stall type represents about 35.8% of the total average of 16.4 cycles between issuing two instructions.

Math Pipe Throttle Stalls Est. Local Speedup: 34.47% On average, each warp of this kernel spends 5.6 cycles being stalled waiting for the execution pipe to be available. This stall occurs when all active warps execute their next instruction on a specific, oversubscribed math pipeline. Try to increase the number of active warps to hide the existent latency or try changing the instruction mix to utilize all available pipelines in a more balanced way. This stall type represents about 34.5% of the total average of 16.4 cycles between issuing two instructions.

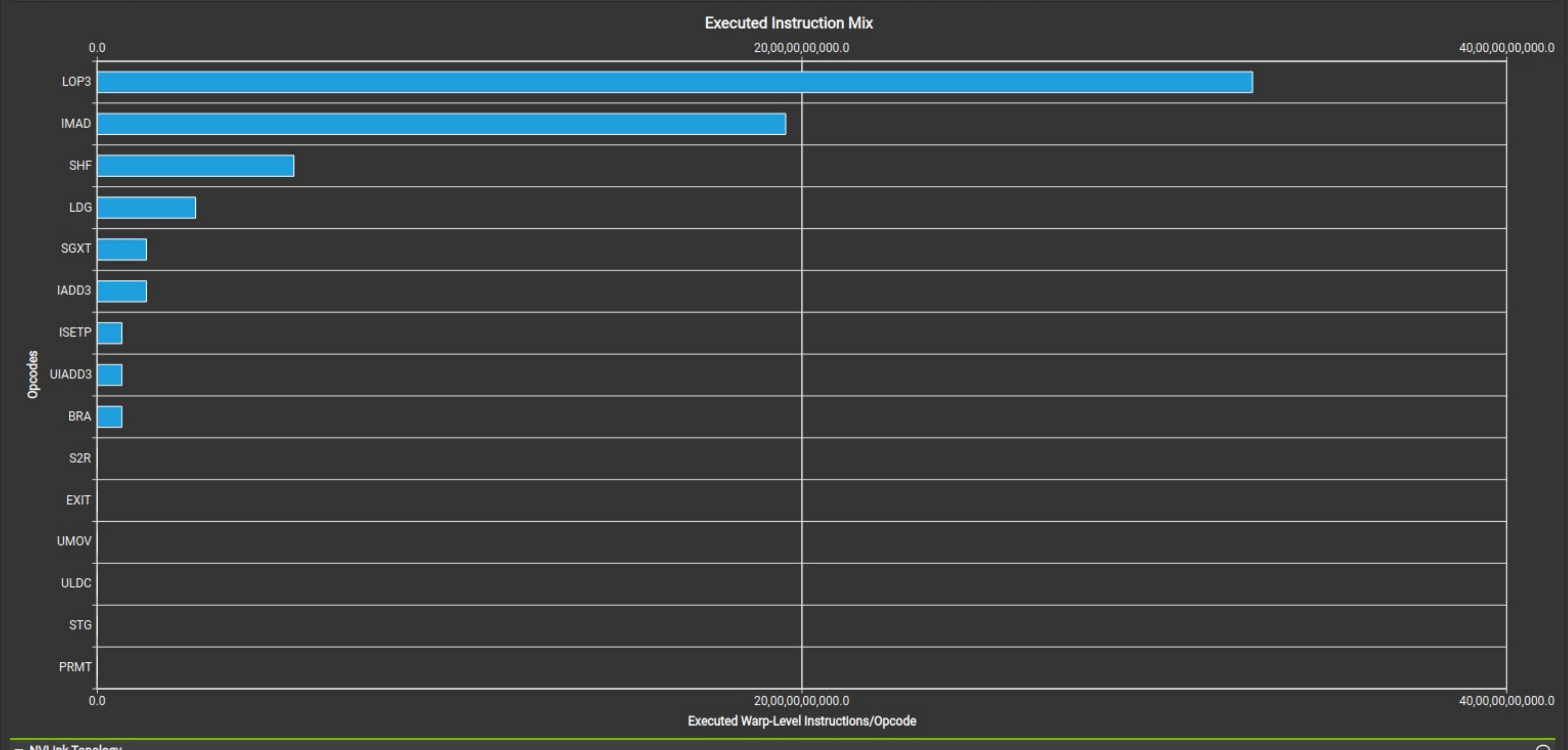
Warp Stall Check the Warp Stall Sampling (All Samples) table for the top stall locations in your source based on sampling data. The Kernel Profiling Guide provides more details on each stall reason.



Instruction Statistics

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]	65,58,48,00,000	Avg. Executed Instructions Per Scheduler [inst]	19,51,92,857.14
Issued Instructions [inst]	65,58,48,18,917	Avg. Issued Instructions Per Scheduler [inst]	19,51,92,913.44



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

GPU ID	GPU Name	CPU Affinity
0	NVIDIA GeForce RTX 3090 Ti	0-15

Launch Statistics

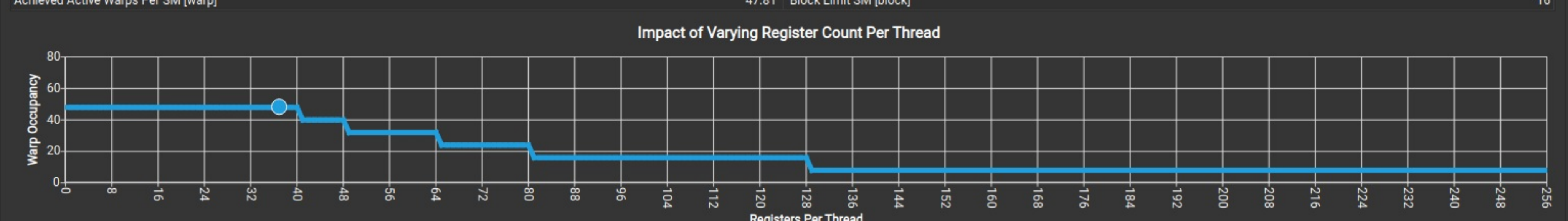
Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.

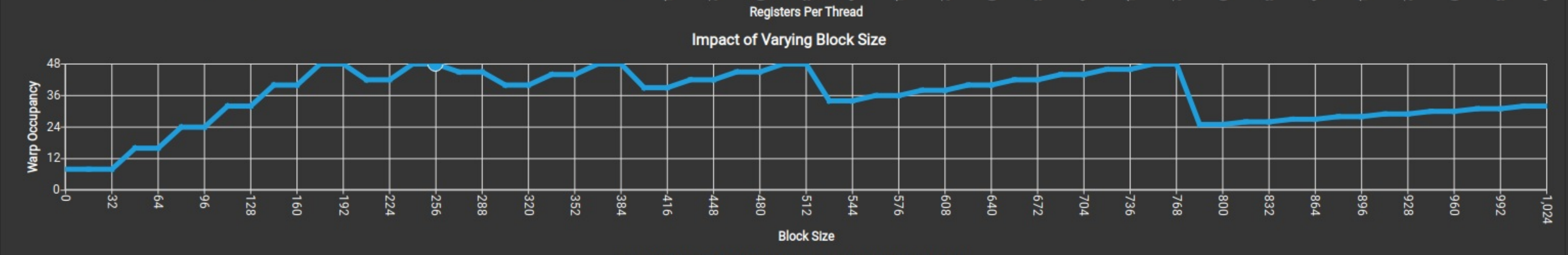
Grid Size	50,000	Function Cache Configuration	CachePreferNone
Registers Per Thread [register/thread]	37	Static Shared Memory Per Block [byte/block]	0
Block Size	256	Dynamic Shared Memory Per Block [byte/block]	0
Threads [thread]	1,28,00,000	Driver Shared Memory Per Block [Kbyte/block]	1.02
Waves Per SM	99.21	Shared Memory Configuration Size [Kbyte]	8.19

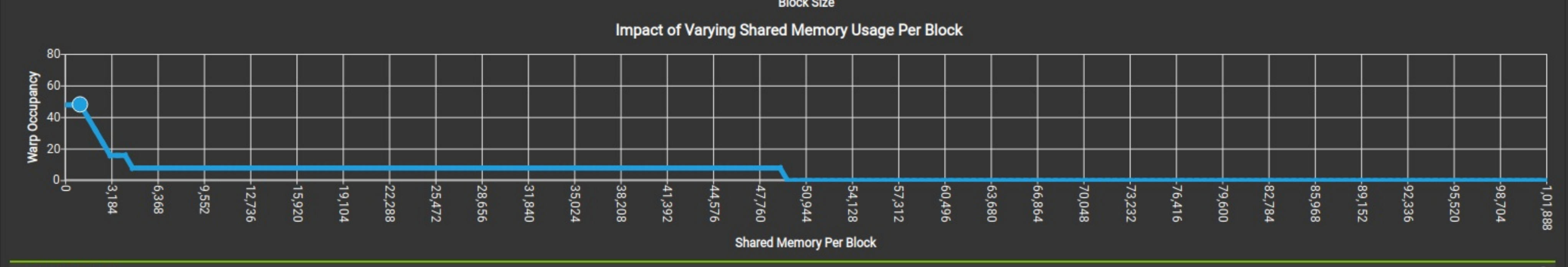
Occupancy

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]	48	Block Limit Shared Mem [block]	8
Achieved Occupancy [%]	99.60	Block Limit Warps [block]	6
Achieved Active Warps Per SM [warp]	47.81	Block Limit SM [block]	16







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]	69,84,00,000	Branch Efficiency [%]	100
Branch Instructions Ratio [%]	0.01	Avg. Divergent Branches	0

Uncoalesced Global Accesses Est. Speedup: 58.54% This kernel has uncoalesced global accesses resulting in a total of 19538400000 excessive sectors (82% of the total 23724800000 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.

L2 Theoretical Sectors Global Excessive

Location	Value	Value (%)
0x7fcd86a57980 in matrixVectorMulKernel	9,76,64,00,000	50
0x7fcd86a57960 in matrixVectorMulKernel	9,76,64,00,000	50
0x7fcd86a57f60 in matrixVectorMulKernel	56,00,000	0
0x7fcd86a579a0 in matrixVectorMulKernel	0	0
0x7fcd86a57990 in matrixVectorMulKernel	0	0

Warp Stall Sampling (All Samples)

Location	Value	Value (%)
0x7fcd86a579f0 in matrixVectorMulKernel	8,62,802	24
0x7fcd86a57da0 in matrixVectorMulKernel	4,12,565	11
0x7fcd86a579c0 in matrixVectorMulKernel	4,06,915	11
0x7fcd86a57b70 in matrixVectorMulKernel	3,39,907	9
0x7fcd86a57a40 in matrixVectorMulKernel	2,67,115	7

Most Instructions Executed

Location	Value	Value (%)
0x7fcd86a57f20 in matrixVectorMulKernel	69,76,00,000	1
0x7fcd86a57f10 in matrixVectorMulKernel	69,76,00,000	1
0x7fcd86a57f00 in matrixVectorMulKernel	69,76,00,000	1
0x7fcd86a57ef0 in matrixVectorMulKernel	69,76,00,000	1
0x7fcd86a57ee0 in matrixVectorMulKernel	69,76,00,000	1

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.