Page: Details | Result: 3 - 133 - d_root | Add Baseline | Apply Rules | Occupancy Calculator | Source Comparison | Save as PDF

	Result	Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current	133 - d_root (14, 1, 1)x(256, 1, 1)	457.06 msecond	71,29,35,473	33	0 - NVIDIA GeForce RTX 3090 Ti	1.56 cycle/nsecond	8.6	[6025] Decrypt_Serial

GPU Speed Of Light Throughput

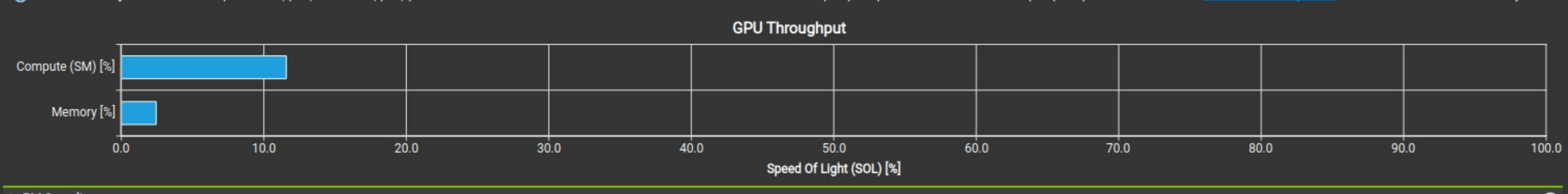
GPU Throughput Chart

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 [%]	11.58	Duration [msecond]	457.06
Memory Throughput [%]	2.44	Elapsed Cycles [cycle]	71,29,35,473
L1/TEX Cache Throughput [%]	14.77	SM Active Cycles [cycle]	11,77,31,513.20
L2 Cache Throughput [%]	0.09	SM Frequency [cycle/nsecond]	1.56
DRAM Throughput [%]	0.16	DRAM Frequency [cycle/nsecond]	10.24

Small Grid This kernel grid is too small to fill the available resources on this device, resulting in only 0.0 full waves across all SMs. Look at Launch Statistics for more details.

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]	0.38	SM Busy [%]	70.10
Executed Ipc Active [inst/cycle]	2.30	Issue Slots Busy [%]	57.51
Issued Ipc Active [inst/cycle]	2.30		

High Utilization ALU is the highest-utilized pipeline (70.1%) based on active cycles, taking into account the rates of its different instructions. It executes integer and logic operations. The pipeline is well-utilized, but might become a bottleneck if more work is added. Based on the number of executed instructions, the highest utilized pipeline (70.1%) 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.

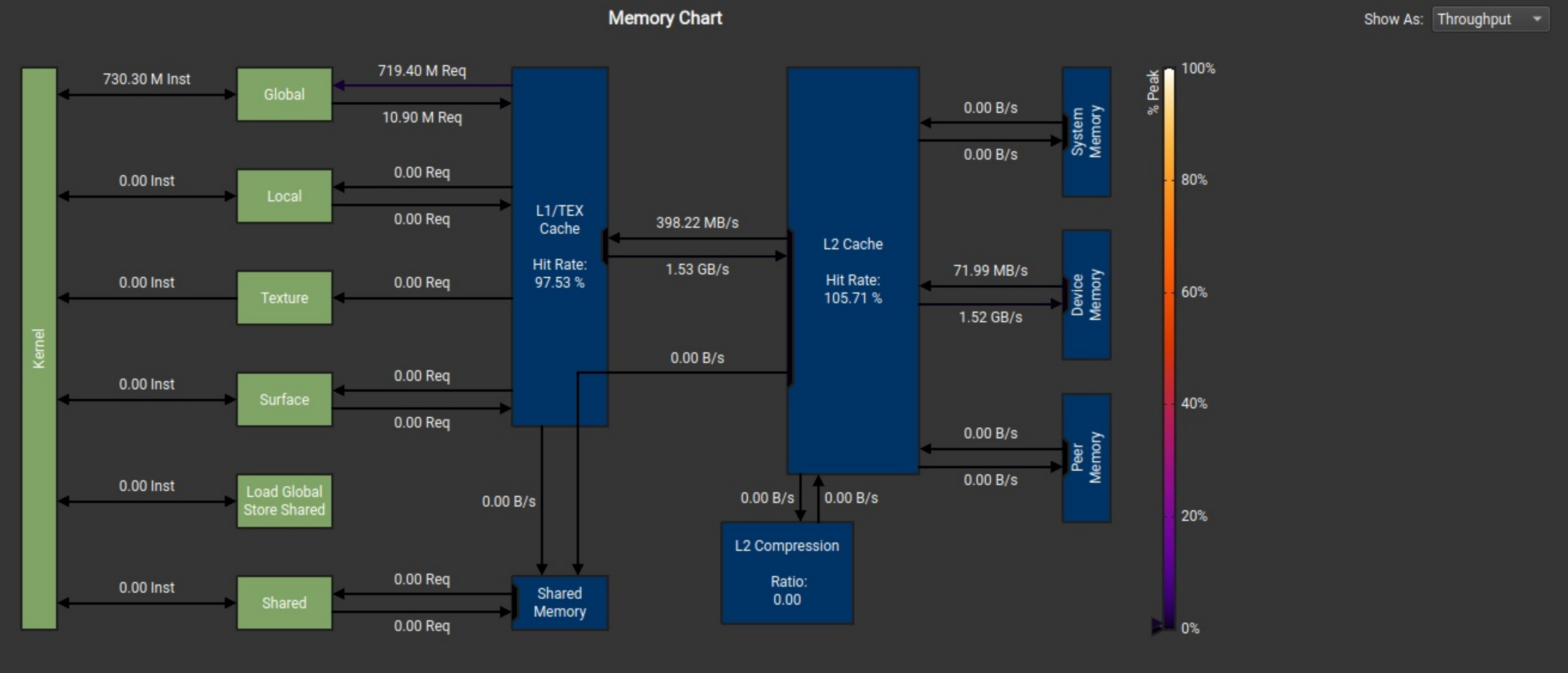
Memory Throughput [Gbyte/second]	1.60	Mem Busy [%]	1.22
L1/TEX Hit Rate [%]	97.53	Max Bandwidth [%]	2.44
L2 Hit Rate [%]	105.71	Mem Pipes Busy [%]	2.44
L2 Compression Success Rate [%]	0	L2 Compression Ratio	0

L2 Load Access Pattern
Est. Speedup: 0.01% 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: 0.02% 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 3.0 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

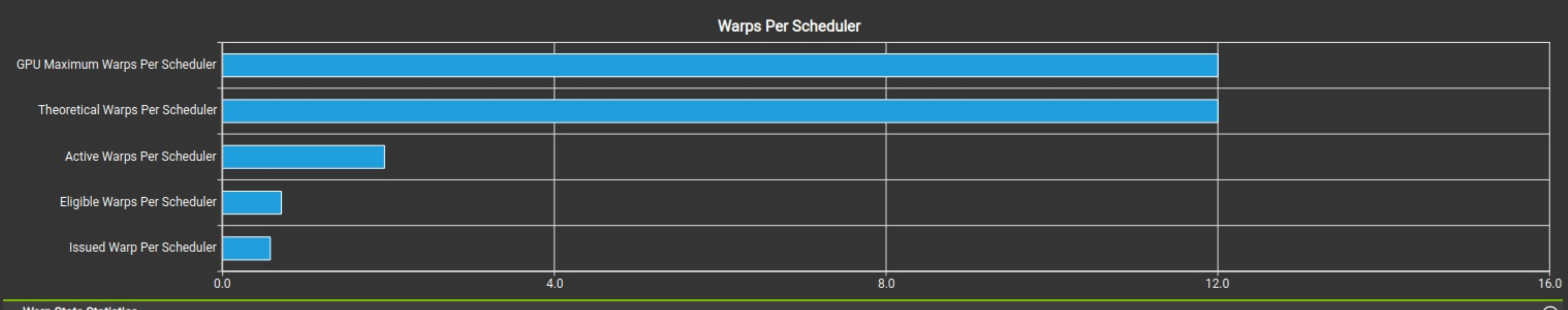


Scheduler Statistics

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]	1.95	No Eligible [%]	42.50
Eligible Warps Per Scheduler [warp]	0.71	One or More Eligible [%]	57.50
Issued Warp Per Scheduler	0.57		

Issue Slot Utilization
Est. Local Speedup: 42.50% Every scheduler is capable of issuing one instruction per cycle, but for this kernel each scheduler only issues an instruction every 1.7 cycles. This might leave hardware resources underutilized and may lead to less optimal performance. Out of the maximum of 12 warps per scheduler, this kernel allocates an average of 1.95 active warps per scheduler, but only an average of 0.71 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, reduce the time the active warps are stalled by inspecting the top stall reasons on the Warp State Statistics and Source Counters sections.



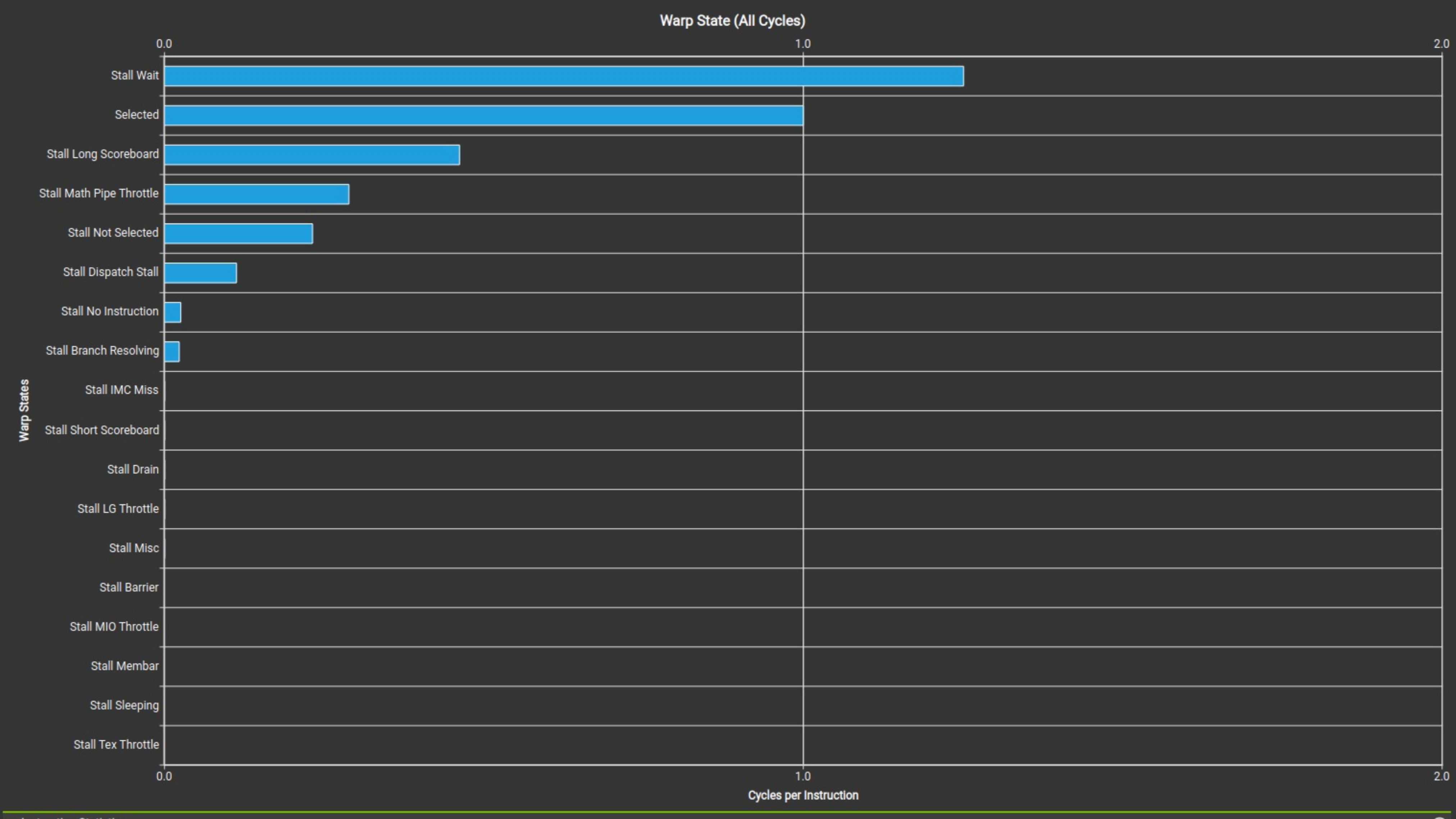
Warp State Statistics

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]	3.40	Avg. Active Threads Per Warp	32
Warp Cycles Per Executed Instruction [cycle]	3.40	Avg. Not Predicated Off Threads Per Warp	31.98

Wait Stalls
Est. Speedup: 36.84% On average, each warp of this kernel spends 1.3 cycles being stalled waiting on a fixed latency execution dependency. Typically, this stall reason should be very low and only shows up as a top contributor in already highly optimized kernels. Try to hide the corresponding instruction latencies by increasing the number of active warps, restructuring the code or unrolling loops. Furthermore, consider switching to lower-latency instructions, e.g. by making use of fast math compiler options. This stall type represents about 36.8% of the total average of 3.4 cycles between issuing two instructions.

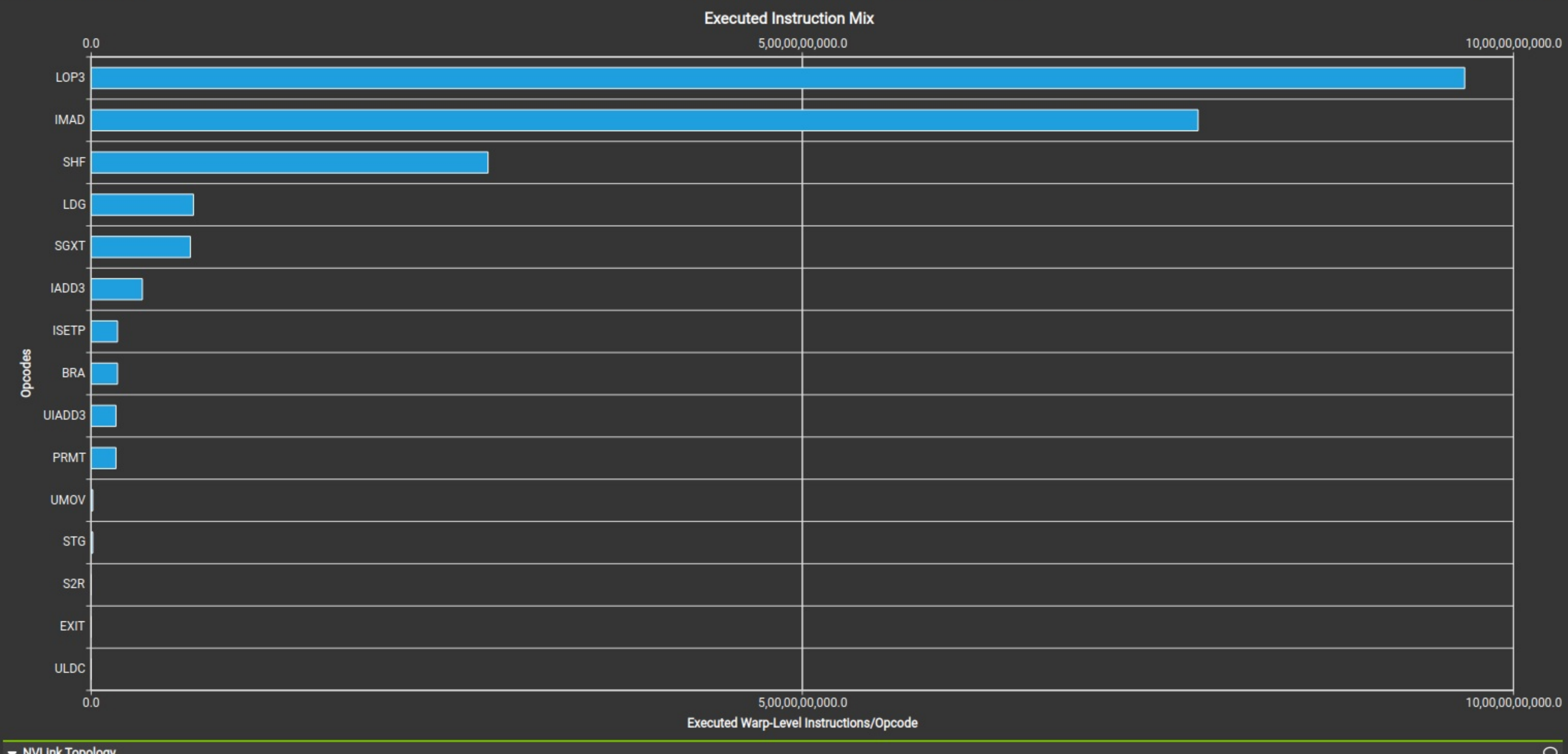
Warp Stall Check the Warp Stall Sampling (All Samples) table for the top stall locations in your source based on sampling data. The Kernel Profiling Guide provides more details on each stall reason.



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]	22,74,83,01,223	Avg. Executed Instructions Per Scheduler [inst]	6,77,03,277.45
Issued Instructions [inst]	22,74,83,02,059	Avg. Issued Instructions Per Scheduler [inst]	6,77,03,279.94



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	14	Function Cache Configuration	CachePreferNone
Registers Per Thread [register/thread]	33	Static Shared Memory Per Block [byte/block]	0
Block Size	256	Dynamic Shared Memory Per Block [byte/block]	0
Threads [thread]	3,584	Driver Shared Memory Per Block [Kbyte/block]	1.02
Waves Per SM	0.03	Shared Memory Configuration Size [Kbyte]	8.19

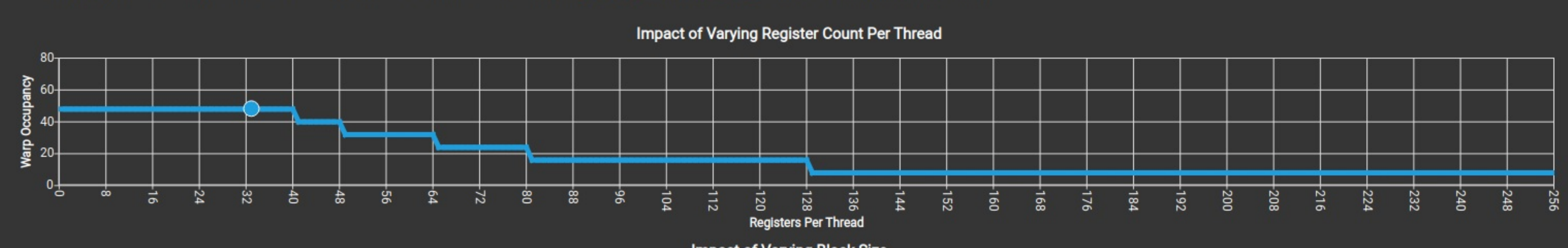
Small Grid
Est. Speedup: 83.33% The grid for this launch is configured to execute only 14 blocks, which is less than the GPU's 84 multiprocessors. This can underutilize some multiprocessors. If you do not intend to execute this kernel concurrently with other workloads, consider reducing the block size to have at least one block per multiprocessor or increase the size of the grid to fully utilize the available hardware resources. See the Hardware Model description for more details on launch configurations.

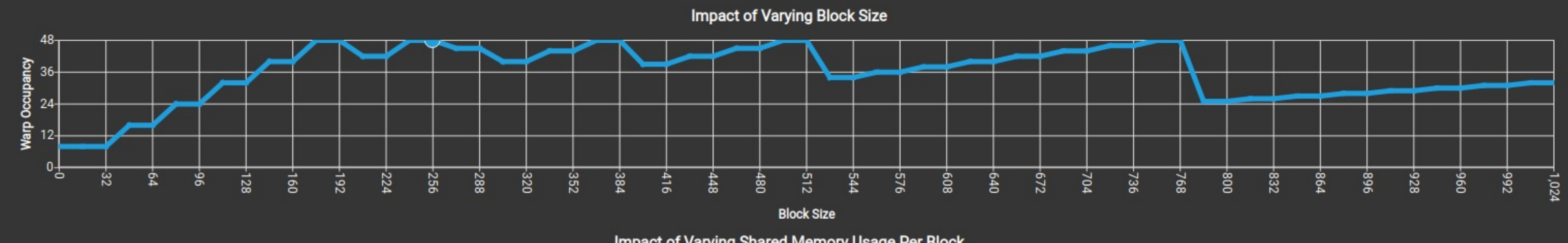
Occupancy

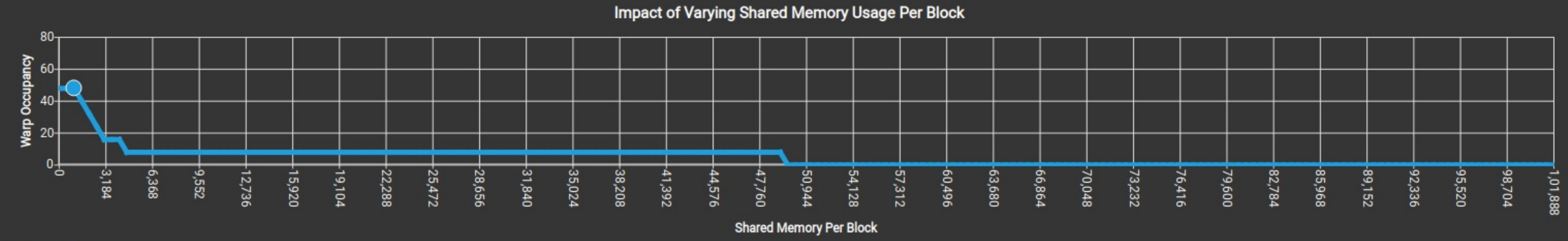
Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, 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 [%]	16.27	Block Limit Warps [block]	6
Achieved Active Warps Per SM [warp]	7.81	Block Limit SM [block]	16

Achieved Occupancy
Est. Speedup: 42.50% The difference between calculated theoretical (100.0%) and measured achieved occupancy (16.3%) 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.







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]	18,53,00,221	Branch Efficiency [%]	100
Branch Instructions Ratio [%]	0.01	Avg. Divergent Branches	0

Warp Stall Sampling (All Samples)

Location	Value	Value (%)
0x7fcd86a5fc10 in d_root	2,06,583	6
0x7fcd86a5f7f0 in d_root	1,55,179	4
0x7fcd86a5fa50 in d_root	1,16,605	3
0x7fcd86a5ffa0 in d_root	97,172	3
0x7fcd86a5fa40 in d_root	92,372	3

Most Instructions Executed

Location	Value	Value (%)
0x7fcd86a5ffb0 in d_root	17,44,00,000	1
0x7fcd86a5ffa0 in d_root	17,44,00,000	1
0x7fcd86a5ff90 in d_root	17,44,00,000	1
0x7fcd86a5ff80 in d_root	17,44,00,000	1
0x7fcd86a5ff70 in d_root	17,44,00,000	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.