	Result	Size	Time	Cycles	GPU	SM Frequency	Process	Attributes
Current	38096 - _5x_cudnn_ampere_fp16_scudnn_fp16_128x64_relu_small_nn_v1	(2747, 1, 1)x(128, 1, 1)	3.39 ms	3,103,434	0 - Orin	914.54 Mhz	[50] python3.10	

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 [%]	74.04	Duration [ms]	3.39
Memory Throughput [%]	46.14	Elapsed Cycles [cycle]	3,103,434
L1/TEX Cache Throughput [%]	46.24	SM Active Cycles [cycle]	3,085,837.12
L2 Cache Throughput [%]	16.59	SM Frequency [Mhz]	914.54

High Compute Throughput Compute is more heavily utilized than Memory: Look at the Compute Workload Analysis section to see what the compute pipelines are spending their time doing. Also, consider whether any computation is redundant and could be reduced or moved to look-up tables.

Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 64:1. The kernel achieved 46% 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 how workload behavior changes over its runtime.

Maximum Sampling Interval [us]	1	# Pass Groups	2
Maximum Buffer Size [Mbyte]	8.39	Dropped Samples [sample]	256

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]	2.96	SM Busy [%]	74.20
Executed Ipc Active [inst/cycle]	2.97	Issue Slots Busy [%]	74.20
Issued Ipc Active [inst/cycle]	2.97		

Balanced FMA is the highest-utilized pipeline (47.4%) based on active cycles, taking into account the rates of its different instructions. It executes 32-bit floating point (FADD, FMUL, FMAD, ...) and integer (IMUL, IMAD) 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.

Mem Busy [%]	46.14	L1/TEX Hit Rate [%]	2.63
Max Bandwidth [%]	33.76	L2 Hit Rate [%]	32.40
Mem Pipes Busy [%]	30.60	L2 Compression Success Rate [%]	0
L2 Compression Ratio	0	-	-

Memory L2 Compression The optional metric dram__bytes_read.sum.pct_of_peak_sustained_elapsed could not be found. Collecting it as an additional metric could enable the rule to provide more guidance.

DRAM Global Load Access Pattern
Est. Local Speedup: 0.98% The memory access pattern for global loads from DRAM might not be optimal. On average, only 31.5 of the 32 bytes transmitted per sector are utilized by each thread. This applies to the 68.6% of sectors missed in L2. This could possibly be caused by a stride between threads. Check the Source Counters section for uncoalesced global loads.

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]	3.97	No Eligible [%]	25.79
Eligible Warps Per Scheduler [warp]	1.65	One or More Eligible [%]	74.21
Issued Warp Per Scheduler	0.74		

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]	5.34	Avg. Active Threads Per Warp	32
Warp Cycles Per Executed Instruction [cycle]	5.34	Avg. Not Predicated Off Threads Per Warp	31.37

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]	73,267,984	Avg. Executed Instructions Per Scheduler [inst]	2,289,624.50
Issued Instructions [inst]	73,270,031	Avg. Issued Instructions Per Scheduler [inst]	2,289,688.47

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	2,747	Function Cache Configuration	CachePreferNone
Registers Per Thread [register/thread]	128	Static Shared Memory Per Block [Kbyte/block]	16.38
Block Size	128	Dynamic Shared Memory Per Block [byte/block]	0
Threads [thread]	351,616	Driver Shared Memory Per Block [Kbyte/block]	1.02
Waves Per SM	85.84	Shared Memory Configuration Size [Kbyte]	102.40
Uses Green Context	0	# SMs [SM]	8

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 [%]	33.33	Block Limit Registers [block]	4
Theoretical Active Warps per SM [warp]	16	Block Limit Shared Mem [block]	5
Achieved Occupancy [%]	33.04	Block Limit Warps [block]	12
Achieved Active Warps Per SM [warp]	15.86	Block Limit SM [block]	16

Theoretical Occupancy
Est. Local Speedup: 66.67% The 4.00 theoretical warps per scheduler this kernel can issue according to its occupancy are below the hardware maximum of 12. This kernel's theoretical occupancy (33.3%) is limited by the number of required registers.

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]	3,085,837.12	Average L1 Active Cycles [cycle]	3,085,837.12
Average L2 Active Cycles [cycle]	3,074,449.25	Average SMSP Active Cycles [cycle]	3,085,287.12
Total SM Elapsed Cycles [cycle]	24,741,240	Total L1 Elapsed Cycles [cycle]	24,741,240
Total L2 Elapsed Cycles [cycle]	12,413,564	Total SMSP Elapsed Cycles [cycle]	98,964,960

Workload Imbalance The optional metric dram__cycles_active.avg could not be found. Collecting it as an additional metric could enable the rule to provide more guidance.

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]	516,436	Branch Efficiency [%]	100
Branch Instructions Ratio [%]	0.01	Avg. Divergent Branches	0

Uncoalesced Global Accesses
Est. Speedup: 0.71% This kernel has uncoalesced global accesses resulting in a total of 10987 excessive sectors (1% of the total 1527265 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 (%)
0x20f3c9020 in _5x_cudnn_ampere_fp16_scudnn_fp16_128x64_relu_small_nn_v1	10,987	100
0x20f3cf010 in _5x_cudnn_ampere_fp16_scudnn_fp16_128x64_relu_small_nn_v1	0	0
0x20f3cf000 in _5x_cudnn_ampere_fp16_scudnn_fp16_128x64_relu_small_nn_v1	0	0
0x20f3cefa0 in _5x_cudnn_ampere_fp16_scudnn_fp16_128x64_relu_small_nn_v1	0	0
0x20f3cef30 in _5x_cudnn_ampere_fp16_scudnn_fp16_128x64_relu_small_nn_v1	0	0

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.