	Result	Size	Time	Cycles	GPU	SM Frequency	Process	Attributes
Current	4579 - CatArrayBatchedCopy_aligned16_contig	(2061, 2, 1)x(128, 1, 1)	22.08 us	29,953	0 - NVIDIA GeForce RTX 3090	1.35 Ghz	[12764] python3.12	

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 [%]	47.24	Duration [us]	22.08
Memory Throughput [%]	78.02	Elapsed Cycles [cycle]	29,953
L1/TEX Cache Throughput [%]	30.38	SM Active Cycles [cycle]	22,538.37
L2 Cache Throughput [%]	38.77	SM Frequency [Ghz]	1.35
DRAM Throughput [%]	78.02	DRAM Frequency [Ghz]	9.72

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.

The following table lists the metrics that are key performance indicators:

Metric Name	Value	Guidance
gpu__compute_memory_throughput.avg.pct_of_peak_sustained_elapsed	78.0238	78.024 - 47.236 >= 10.000

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 how workload behavior changes over its runtime.

Maximum Sampling Interval [us]	1	# Pass Groups	2
Maximum Buffer Size [Mbytes]	8	Dropped Samples [sample]	0

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]	1.77	SM Busy [%]	55.29
Executed Ipc Active [inst/cycle]	2.07	Issue Slots Busy [%]	52.10
Issued Ipc Active [inst/cycle]	2.08		

Balanced ALU is the highest-utilized pipeline (44.6%) based on active cycles, 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

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]	728.33	Mem Busy [%]	38.77
L1/TEX Hit Rate [%]	60.00	Max Bandwidth [%]	78.02
L2 Hit Rate [%]	50.81	Mem Pipes Busy [%]	9.96
L2 Compression Success Rate [%]	0	L2 Compression Ratio	0

L1TEX Global Store Access Pattern
Est. Speedup: 22.79% The memory access pattern for global stores to L1TEX might not be optimal. On average, only 8.0 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.

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]	9.62	No Eligible [%]	46.14
Eligible Warps Per Scheduler [warp]	2.97	One or More Eligible [%]	53.86
Issued Warp Per Scheduler	0.54		

Issue Slot Utilization
Est. Local Speedup: 21.98% Every scheduler is capable of issuing one instruction per cycle, but for this kernel each scheduler only issues an instruction every 1.9 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 9.62 active warps per scheduler, but only an average of 2.97 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.

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]	17.86	Avg. Active Threads Per Warp	32.00
Warp Cycles Per Executed Instruction [cycle]	17.99	Avg. Not Predicated Off Threads Per Warp	26.62

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]	3,823,890	Avg. Executed Instructions Per Scheduler [inst]	11,658.20
Issued Instructions [inst]	3,851,681	Avg. Issued Instructions Per Scheduler [inst]	11,742.93

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	4,122	Function Cache Configuration	CachePreferNone
Registers Per Thread [register/thread]	38	Static Shared Memory Per Block [byte/block]	0
Block Size	128	Dynamic Shared Memory Per Block [byte/block]	0
Threads [thread]	527,616	Driver Shared Memory Per Block [Kbyte/block]	1.02
Waves Per SM	4.19	Shared Memory Configuration Size [Kbyte]	16.38
Uses Green Context	0	# SMs [SM]	82

Tail Effect
Est. Speedup: 20.00% 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 4 full waves and a partial wave of 185 thread blocks. Under the assumption of a uniform execution duration of all thread blocks, the partial wave may account for up to 20.0% of the total kernel runtime with a lower occupancy of 22.2%. 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.

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]	12
Theoretical Active Warps per SM [warp]	48	Block Limit Shared Mem [block]	16
Achieved Occupancy [%]	77.78	Block Limit Warps [block]	12
Achieved Active Warps Per SM [warp]	37.33	Block Limit SM [block]	16

Achieved Occupancy
Est. Speedup: 21.98% The difference between calculated theoretical (100.0%) and measured achieved occupancy (77.8%) 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.

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]	22,538.37	Average L1 Active Cycles [cycle]	22,538.37
Average L2 Active Cycles [cycle]	25,267.96	Average SMSP Active Cycles [cycle]	21,801.72
Average DRAM Active Cycles [cycle]	167,516	Total SM Elapsed Cycles [cycle]	2,163,384
Total L1 Elapsed Cycles [cycle]	2,163,384	Total L2 Elapsed Cycles [cycle]	1,462,512
Total SMSP Elapsed Cycles [cycle]	8,653,536	Total DRAM Elapsed Cycles [cycle]	2,576,384

SMs Workload Imbalance
Est. Speedup: 5.68% One or more SMs have a much higher number of active cycles than the average number of active cycles. Additionally, other SMs have a much lower number of active cycles than the average number of active cycles. Maximum instance value is 6.65% above the average, while the minimum instance value is 6.57% below the average.

SMSPs Workload Imbalance
Est. Speedup: 7.65% One or more SMSPs have a much higher number of active cycles than the average number of active cycles. Additionally, other SMSPs have a much lower number of active cycles than the average number of active cycles. Maximum instance value is 9.25% above the average, while the minimum instance value is 8.88% below the average.

L1 Slices Workload Imbalance
Est. Speedup: 5.68% One or more L1 Slices have a much higher number of active cycles than the average number of active cycles. Additionally, other L1 Slices have a much lower number of active cycles than the average number of active cycles. Maximum instance value is 6.65% above the average, while the minimum instance value is 6.57% below the average.

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]	82,416	Branch Efficiency [%]	100
Branch Instructions Ratio [%]	0.02	Avg. Divergent Branches	0

Uncoalesced Global Accesses
Est. Speedup: 49.76% This kernel has uncoalesced global accesses resulting in a total of 791096 excessive sectors (60% of the total 1318500 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 (%)
0x75d88f25ff30 in CatArrayBatchedCopy_aligned16_contig	197,774	25
0x75d88f25ff20 in CatArrayBatchedCopy_aligned16_contig	197,774	25
0x75d88f25ff00 in CatArrayBatchedCopy_aligned16_contig	197,774	25
0x75d88f25fee0 in CatArrayBatchedCopy_aligned16_contig	197,774	25
0x75d88f261290 in CatArrayBatchedCopy_aligned16_contig	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.