	Result	Size	Time	Cycles	GPU	SM Frequency	Process	Attributes
Current	574 - brute7	(1024, 1, 1)x(256, 1, 1)	53.34 us	47,985	0 - NVIDIA GeForce RTX 3060 Laptop GPU	899.35 Mhz	[2152] gpu_crack.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 [%]	50.48	Duration [us]	53.34
Memory Throughput [%]	52.00	Elapsed Cycles [cycle]	47,985
L1/TEX Cache Throughput [%]	81.41	SM Active Cycles [cycle]	42,646.70
L2 Cache Throughput [%]	47.93	SM Frequency [Mhz]	899.35
DRAM Throughput [%]	52.00	DRAM Frequency [Ghz]	6.98

Latency Issue This kernel 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.

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]	1	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]	1.61	SM Busy [%]	51.58
Executed Ipc Active [inst/cycle]	1.65	Issue Slots Busy [%]	41.53
Issued Ipc Active [inst/cycle]	1.66		

Balanced ALU is the highest-utilized pipeline (51.6%) 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

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]	172.11	Mem Busy [%]	47.93
L1/TEX Hit Rate [%]	67.41	Max Bandwidth [%]	52.00
L2 Hit Rate [%]	95.56	Mem Pipes Busy [%]	37.29
L2 Compression Success Rate [%]	0	L2 Compression Ratio	0

L1TEX Global Load Access Pattern
Est. Speedup: 50.82%
The memory access pattern for global loads from L1TEX might not be optimal. On average, only 12.0 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 loads.

L1TEX Global Store Access Pattern
Est. Speedup: 61.06%
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.

L1TEX Local Load Access Pattern
Est. Speedup: 79.62%
The memory access pattern for local loads from L1TEX might not be optimal. On average, only 0.7 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 local loads.

L1TEX Local Store Access Pattern
Est. Speedup: 80.18%
The memory access pattern for local stores to L1TEX might not be optimal. On average, only 0.5 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 local 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]	11.32	No Eligible [%]	59.59
Eligible Warps Per Scheduler [warp]	1.50	One or More Eligible [%]	40.41
Issued Warp Per Scheduler	0.40		

Issue Slot Utilization
Est. Local Speedup: 48.00%
Every scheduler is capable of issuing one instruction per cycle, but for this kernel each scheduler only issues an instruction every 2.5 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 11.32 active warps per scheduler, but only an average of 1.50 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]	28.02	Avg. Active Threads Per Warp	31.11
Warp Cycles Per Executed Instruction [cycle]	28.28	Avg. Not Predicated Off Threads Per Warp	30.57

Long Scoreboard Stalls
Est. Speedup: 34.91%
On average, each warp of this kernel spends 9.8 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 34.9% of the total average of 28.0 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]	2,105,189	Avg. Executed Instructions Per Scheduler [inst]	17,543.24
Issued Instructions [inst]	2,125,319	Avg. Issued Instructions Per Scheduler [inst]	17,710.99

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	1,024	Function Cache Configuration	CachePreferNone
Registers Per Thread [register/thread]	40	Static Shared Memory Per Block [byte/block]	0
Block Size	256	Dynamic Shared Memory Per Block [byte/block]	0
Threads [thread]	262,144	Driver Shared Memory Per Block [Kbyte/block]	1.02
Waves Per SM	5.69	Shared Memory Configuration Size [Kbyte]	8.19
Uses Green Context	0	# SMs [SM]	30

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 [%]	95.91	Block Limit Warps [block]	6
Achieved Active Warps Per SM [warp]	46.03	Block Limit SM [block]	16

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]	42,646.70	Average L1 Active Cycles [cycle]	42,646.70
Average L2 Active Cycles [cycle]	36,744.67	Average SMSP Active Cycles [cycle]	43,828.82
Average DRAM Active Cycles [cycle]	191,272	Total SM Elapsed Cycles [cycle]	1,307,310
Total L1 Elapsed Cycles [cycle]	1,307,310	Total L2 Elapsed Cycles [cycle]	1,103,592
Total SMSP Elapsed Cycles [cycle]	5,229,240	Total DRAM Elapsed Cycles [cycle]	2,235,392

L2 Slices Workload Imbalance
Est. Speedup: 7.95%
One or more L2 Slices have a much higher number of active cycles than the average number of active cycles. Maximum instance value is 9.95% above the average, while the minimum instance value is 2.95% 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]	85,530	Branch Efficiency [%]	95.99
Branch Instructions Ratio [%]	0.04	Avg. Divergent Branches	19.08

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.