Page: Details | Result: 0 - 557 - calculateHisto | Add Baseline | Apply Rules | Occupancy Calculator | Save as PDF

	Result	Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current	557 - calculateHisto (3125000, 1, 1)x(32, 1, 1)	37.53 msecond	49,307,060	32	0 - NVIDIA GeForce RTX 3060	1.31 cycle/nsecond	8.6	[13588] AsssignmentStrategy1.exe

GPU Speed Of Light Throughput

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 [%]	18.21	Duration [msecond]	37.53
Memory Throughput [%]	14.20	Elapsed Cycles [cycle]	49,307,060
L1/TEX Cache Throughput [%]	17.08	SM Active Cycles [cycle]	48,558,004.18
L2 Cache Throughput [%]	14.20	SM Frequency [cycle/nsecond]	1.31
DRAM Throughput [%]	1.03	DRAM Frequency [cycle/nsecond]	7.26

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.

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]	0.51	SM Busy [%]	18.49
Executed Ipc Active [inst/cycle]	0.52	Issue Slots Busy [%]	12.98
Issued Ipc Active [inst/cycle]	0.52		

Low Utilization All compute pipelines are under-utilized. Either this kernel is very small or it doesn't issue enough warps per scheduler. Check the Launch Statistics and Scheduler Statistics sections for further details.

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/second]	3.59	Mem Busy [%]	11.06
L1/TEX Hit Rate [%]	79.22	Max Bandwidth [%]	14.20
L2 Hit Rate [%]	95.58	Mem Pipes Busy [%]	6.33
L2 Compression Success Rate [%]	0	L2 Compression Ratio	0

L1TEX Local Load Access Pattern The memory access pattern for local loads in L1TEX might not be optimal. On average, this kernel accesses 3.7 bytes per thread per memory request; but the address pattern, possibly caused by the stride between threads, results in 3.9 sectors per request, or 3.9*32 = 123.7 bytes of cache data transfers per request. The optimal thread address pattern for 3.7 byte accesses would result in 3.7*32 = 120.0 bytes of cache data transfers per request, to maximize L1TEX cache performance. Check the Source Counters section for uncoalesced local loads.

L1TEX Local Store Access Pattern The memory access pattern for local stores in L1TEX might not be optimal. On average, this kernel accesses 3.7 bytes per thread per memory request; but the address pattern, possibly caused by the stride between threads, results in 3.9 sectors per request, or 3.9*32 = 123.7 bytes of cache data transfers per request. The optimal thread address pattern for 3.7 byte accesses would result in 3.7*32 = 120.0 bytes of cache data transfers per request, to maximize L1TEX cache performance. Check the Source Counters section for uncoalesced local stores.

L2 Load Access Pattern 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 1.1 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.

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]	3.80	No Eligible [%]	86.72
Eligible Warps Per Scheduler [warp]	0.14	One or More Eligible [%]	13.28
Issued Warp Per Scheduler	0.13		

Issue Slot Utilization Every scheduler is capable of issuing one instruction per cycle, but for this kernel each scheduler only issues an instruction every 7.5 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 3.80 active warps per scheduler, but only an average of 0.14 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.

Issue Slot Utilization The 4.00 theoretical warps per scheduler this kernel can issue according to its occupancy are below the hardware maximum of 12. Use the Occupancy section to identify what limits this kernel's theoretical occupancy.

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]	28.62	Avg. Active Threads Per Warp	28.54
Warp Cycles Per Executed Instruction [cycle]	28.62	Avg. Not Predicated Off Threads Per Warp	28.06

wait On average, each warp of this kernel spends 13.5 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 47.2% of the total average of 28.6 cycles between issuing two instructions.

long_scoreboard On average, each warp of this kernel spends 9.1 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 31.7% of the total average of 28.6 cycles between issuing two instructions.

Warp Stall Check the Warp Stall Sampling (All Cycles) 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]	705,797,872	Avg. Executed Instructions Per Scheduler [inst]	6,301,766.71
Issued Instructions [inst]	705,802,134	Avg. Issued Instructions Per Scheduler [inst]	6,301,804.77

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	3,125,000	Function Cache Configuration	CachePreferNone
Registers Per Thread [register/thread]	32	Static Shared Memory Per Block [byte/block]	0
Block Size	32	Dynamic Shared Memory Per Block [byte/block]	0
Threads [thread]	100,000,000	Driver Shared Memory Per Block [Kbyte/block]	1.02
Waves Per SM	6,975.45	Shared Memory Configuration Size [Kbyte]	16.38

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]	64
Theoretical Active Warps per SM [warp]	16	Block Limit Shared Mem [block]	16
Achieved Occupancy [%]	31.12	Block Limit Warps [block]	48
Achieved Active Warps Per SM [warp]	14.94	Block Limit SM [block]	16

Occupancy Limiters This kernel's theoretical occupancy (33.3%) is limited by the number of blocks that can fit on the SM. This kernel's theoretical occupancy (33.3%) is limited by the required amount of shared memory. 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]	84,304,790	Branch Efficiency [%]	90.48
Branch Instructions Ratio	0.12	Avg. Divergent Branches	55,706.77

Uncoalesced Global Accesses This kernel has uncoalesced global accesses resulting in a total of 1046303 excessive sectors (8% of the total 13407645 sectors). Check the L2 Theoretical Sectors Global Excessive table for the primary source locations. The CUDA Programming Guide had additional information on reducing uncoalesced device memory accesses.

L2 Theoretical Sectors Global Excessive

Location	Value	Value (%)
0xd00c62690 in __iAtomicAdd	1,046,303	100

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.