Page: Details | Launch: 0 - 116 - clear_buffer | Add Baseline | Apply Rules | Occupancy Calculator | Save as PDF

	Launch	Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current	116 - clear_buffer (7579, 1, 1)x(128, 1, 1)	42.82 usecond	47,301	16	0 - NVIDIA GeForce RTX 2070 with Max-Q Design	1.10 cycle/nsecond	7.5	[7283] simple

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 [%]	7.14	Duration [usecond]	42.82
Memory Throughput [%]	89.49	Elapsed Cycles [cycle]	47,301
L1/TEX Cache Throughput [%]	57.10	SM Active Cycles [cycle]	43,422.94
L2 Cache Throughput [%]	34.18	SM Frequency [cycle/nsecond]	1.10
DRAM Throughput [%]	89.49	DRAM Frequency [cycle/nsecond]	6.07

High Throughput The kernel is utilizing greater than 80.0% of the available compute or memory performance of the device. To further improve performance, work will likely need to be shifted from the most utilized to another unit. Start by analyzing workloads in the Memory Workload Analysis section.

Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 32:1. The kernel achieved 0% of this device's fp32 peak performance and 0% of its fp64 peak performance. See the Kernel Profiling Guide for mode 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]	0.18	SM Busy [%]	4.90
Executed Ipc Active [inst/cycle]	0.19	Issue Slots Busy [%]	4.90
Issued Ipc Active [inst/cycle]	0.20		

Low Utilization All 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).

Memory Throughput [Gbyte/second]	347.41	Mem Busy [%]	34.18
L1/TEX Hit Rate [%]	0	Max Bandwidth [%]	89.49
L2 Hit Rate [%]	99.98	Mem Pipes Busy [%]	7.14

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]	6.10	No Eligible [%]	95.08
Eligible Warps Per Scheduler [warp]	0.06	One or More Eligible [%]	4.92
Issued Warp Per Scheduler	0.05		

Issue Slot Utilization Every scheduler is capable of issuing one instruction per cycle, but for this kernel each scheduler only issues an instruction every 20.3 cycles. This might leave hardware resources underutilized and may lead to less optimal performance. Out of the maximum of 8 warps per scheduler, this kernel allocates an average of 6.10 active warps per scheduler, but only an average of 0.06 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]	123.83	Avg. Active Threads Per Warp	32
Warp Cycles Per Executed Instruction [cycle]	125.24	Avg. Not Predicated Off Threads Per Warp	28.80

drain On average, each warp of this kernel spends 59.0 cycles being stalled waiting after an EXIT instruction for all outstanding memory instructions to complete so that the warp's resources can be freed. This represents about 47.7% of the total average of 123.8 cycles between issuing two instructions. A high number of stalls due to draining warps typically occurs when a lot of data is written to memory towards the end of a kernel. Make sure the memory access patterns of these store operations are optimal for the target architecture and consider parallelized data reduction, if applicable.

lg_throttle On average, each warp of this kernel spends 41.2 cycles being stalled waiting for the local/global instruction queue to be not full. This represents about 33.3% of the total average of 123.8 cycles between issuing two instructions. Typically this stall occurs only when executing local or global memory instructions extremely frequently. If applicable, consider combining multiple lower-width memory operations into fewer wider memory operations and try interleaving memory operations and math instructions.

Warp Stall Check the Source Counters section 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]	303,160	Avg. Executed Instructions Per Scheduler [inst]	2,105.28
Issued Instructions [inst]	306,600	Avg. Issued Instructions Per Scheduler [inst]	2,129.17

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	7,579	Registers Per Thread [register/thread]	16
Block Size	128	Static Shared Memory Per Block [byte/block]	0
Threads [thread]	970,112	Dynamic Shared Memory Per Block [byte/block]	0
Waves Per SM	26.32	Driver Shared Memory Per Block [byte/block]	0
Function Cache Configuration	cudaFuncCachePreferNone	Shared Memory Configuration Size [Kbyte]	32.77

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]	32
Theoretical Active Warps per SM [warp]	32	Block Limit Shared Mem [block]	16
Achieved Occupancy [%]	76.84	Block Limit Warps [block]	8
Achieved Active Warps Per SM [warp]	24.59	Block Limit SM [block]	16

Occupancy Limiters This kernel's theoretical occupancy is not impacted by any block limit. The difference between calculated theoretical (100.0%) and measured achieved occupancy (76.8%) 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.

Source Counters

Source metrics, including branch efficiency and sampled warp stall reasons. Sampling Data 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]	60,632	Branch Efficiency [%]	0
Branch Instructions Ratio	0.20	Avg. Divergent Branches	0