Result	Size	Time	Cycles	GPU	SM Frequency	Process	Attributes
Current	(32, 32, 1)x(16, 16, 1)	111.79 ms	181,105,143	0 - NVIDIA GeForce RTX 4060 Laptop GPU	1.62 Ghz	[36188] gemm_fp64	
574 - dgemm_fp64							

Summary | Details | Source | Context | Comments | Raw | Session

Compare | Tools | View | Export

GPU Speed Of Light Throughput

Roofline Double Precision

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 [%]	98.91	Duration [ms]	111.79
Memory Throughput [%]	8.59	Elapsed Cycles [cycle]	181,105,143
L1/TEX Cache Throughput [%]	8.65	SM Active Cycles [cycle]	179,696,493.71
L2 Cache Throughput [%]	1.26	SM Frequency [ghz]	1.62
DRAM Throughput [%]	1.68	DRAM Frequency [ghz]	7.99

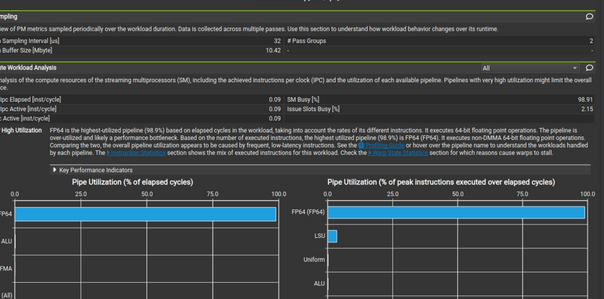
High Throughput This workload is utilizing greater than 80.0% of the available compute or memory performance of the device. To further improve performance, work will likely need to be shifted from the most utilized to another unit. Start by analyzing workloads in the Compute Workload Analysis section.

FP64/32 Utilization The ratio of peak float (FP32) to double (FP64) performance on this device is 64:1. The workload achieved 0% of this device's FP32 peak performance and 99% of its FP64 peak performance. If Compute Workload Analysis determines that this workload is FP64 bound, consider using 32-bit precision floating point operations to improve its performance. See the Profiling Guide for more details on roofline analysis.

Est. Speedup: 97.37%

Key Performance Indicators

Floating Point Operations Roofline (Double Precision)



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]	32	# Pass Groups	2
Maximum Buffer Size [Mbyte]	10.42	-	

Compute Workload Analysis

All

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.

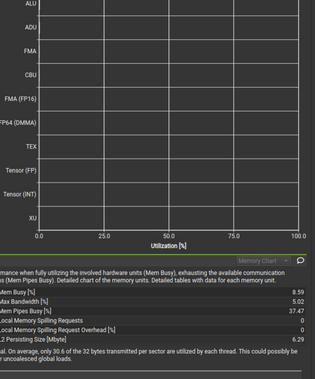
Executed Ipc Elapsed [inst/cycle]	0.09	SM Busy [%]	98.91
Executed Ipc Active [inst/cycle]	0.09	Issue Slots Busy [%]	2.15
Issued Ipc Active [inst/cycle]	0.09		

Very High Utilization FP64 is the highest-utilized pipeline (98.9%) based on elapsed cycles in the workload, taking into account the rates of its different instructions. It executes 64-bit floating point operations. The pipeline is over-utilized and likely a performance bottleneck. Based on the number of executed instructions, the highest utilized pipeline (98.9%) is FP64 (FP64). It executes non-DMMA 64-bit floating point operations. Comparing the two, the overall pipeline utilization appears to be caused by frequent, low-latency instructions. See the Profiling Guide or hover over the pipeline name to understand the workloads handled by each pipeline. The Instruction Statistics section shows the mix of executed instructions for this workload. Check the Warp State Statistics section for which reasons cause warps to stall.

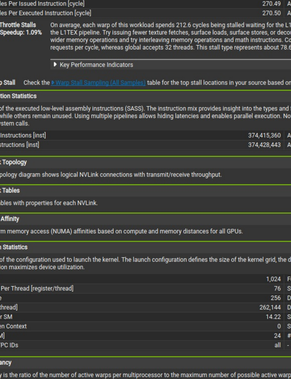
Key Performance Indicators

Pipe Utilization (% of elapsed cycles)

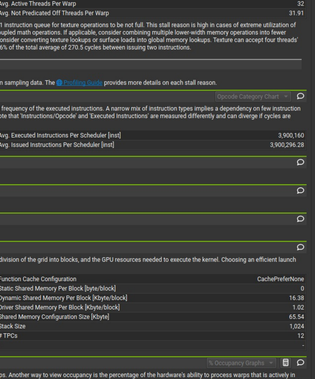
Pipe Utilization (% of peak instructions executed over elapsed cycles)



Pipe Utilization (% of active cycles)



Pipe Utilization (% of peak instructions executed over active cycles)



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]	4.29	Mem Busy [%]	8.59
L1/TEX Hit Rate [%]	9.24	Max Bandwidth [%]	5.02
L2 Hit Rate [%]	80.76	Mem Pipes Busy [%]	37.47
L2 Compression Input Sectors [sector]	0	Local Memory Spilling Requests	0
L2 Compression Ratio	0	Local Memory Spilling Request Overhead [%]	0
L2 Compression Success Rate [%]	0	L2 Persisting Size [Mbyte]	6.29

L1TEX Global Load Access Pattern The memory access pattern for global loads from 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 Memory Counters section for uncoalesced global loads.

Est. Speedup: 0.38%

Key Performance Indicators

L1TEX Global Store Access Pattern 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 Memory Counters section for uncoalesced global stores.

Est. Speedup: 6.44%

Key Performance Indicators

Shared Load Bank Conflicts The memory access pattern for shared loads might not be optimal and causes on average a 5.0 - way bank conflict across all 67108864 shared load requests.This results in 134217728 bank conflicts, which represent 40.00% of the overall 335547196 wavefronts for shared loads. Check the Memory Counters section for uncoalesced shared loads.

Est. Speedup: 3.46%

Key Performance Indicators

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]	5.87	No Eligible [%]	97.83
Eligible Warps Per Scheduler [warp]	0.08	One or More Eligible [%]	2.17
Issued Warps Per Scheduler	0.02		

Issue Slot Utilization On average, each warp of this workload spends 21.6 cycles being stalled waiting for the L1 instruction queue for texture operations to be not full. This stall reason is high in cases of extreme utilization of the L1TEX pipeline. Try issuing fewer texture fetches, surface loads, surface stores, or decoupled math operations. If applicable, consider combining multiple lower-width memory operations into fewer wider memory operations and try interleaving memory operations and math instructions. Consider converting texture lookups or surface loads into global memory lookups. Texture can accept four threads' requests per cycle, whereas global accepts 32 threads. This stall type represents about 78.6% of the total average of 27.6 cycles between issuing two instructions.

Est. Local Speedup: 1.09%

Every scheduler is capable of issuing one instruction per cycle, but for this workload each scheduler only issues an instruction every 46.1 cycles. This might leave hardware resources underutilized and may lead to less optimal performance. Out of the maximum of 12 warps per scheduler, this workload allocates an average of 5.87 active warps per scheduler, but only an average of 0.08 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.

Key Performance Indicators

The Throttle Stalls On average, each warp of this workload spends 213.6 cycles being stalled waiting for the L1 instruction queue for texture operations to be not full. This stall reason is high in cases of extreme utilization of the L1TEX pipeline. Try issuing fewer texture fetches, surface loads, surface stores, or decoupled math operations. If applicable, consider combining multiple lower-width memory operations into fewer wider memory operations and try interleaving memory operations and math instructions. Consider converting texture lookups or surface loads into global memory lookups. Texture can accept four threads' requests per cycle, whereas global accepts 32 threads. This stall type represents about 78.6% of the total average of 27.6 cycles between issuing two instructions.

Est. Speedup: 1.09%

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]	270.49	Avg. Active Threads Per Warp	32
Warp Cycles Per Executed Instruction [cycle]	270.50	Avg. Not Predicated Off Threads Per Warp	31.91

Warp Stall Check the Warp Stall Sampling (All Samples) table for the top stall locations in your source based on sampling data. The Profiling Guide provides more details on each stall reason.

Instruction Statistics

Show Source Counters Chart

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]	374,415,360	Avg. Executed Instructions Per Scheduler [inst]	3,900,160
Issued Instructions [inst]	374,428,443	Avg. Issued Instructions Per Scheduler [inst]	3,900,296.28

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]	76	Static Shared Memory Per Block [byte/block]	0
Block Size	256	Dynamic Shared Memory Per Block [Kbyte/block]	16.38
Threads [thread]	262,144	Driver Shared Memory Per Block [Kbyte/block]	1.02
Waves Per SM	14.22	Shared Memory Configuration Size [Kbyte]	65.54
Uses Green Context	0	Stack Size	1,024
# SMs [SM]	24	# TPCs	12
Enabled TPC IDs	all	-	

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 [%]	50	Block Limit Registers [block]	3
Theoretical Active Warps per SM [warp]	24	Block Limit Shared Mem [block]	3
Achieved Occupancy [%]	48.93	Block Limit Warps [block]	6
Achieved Active Warps Per SM [warp]	23.48	Block Limit SM [block]	24

Theoretical Occupancy The 6.00 theoretical warps per scheduler this kernel can issue according to its occupancy are below the hardware maximum of 12. This kernel's theoretical occupancy (50.0%) is limited by the number of required registers, and the required amount of shared memory.

Est. Speedup: 1.09%

Key Performance Indicators

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]	179,696,493.71	Average L1 Active Cycles [cycle]	179,696,493.71
Average L2 Active Cycles [cycle]	21,084,717.84	Average SMSP Active Cycles [cycle]	179,667,418.33
Average DRAM Active Cycles [cycle]	14,978,168	Total SM Elapsed Cycles [cycle]	4,346,465,632
Total L1 Elapsed Cycles [cycle]	4,346,465,632	Total L2 Elapsed Cycles [cycle]	2,870,031,360
Total SMSP Elapsed Cycles [cycle]	17,385,862,528	Total DRAM Elapsed Cycles [cycle]	3,574,259,712

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]	2,113,536	Branch Efficiency [%]	100
Branch Instructions Ratio [%]	0.01	Avg. Divergent Branches [branches]	0

Uncoalesced Global Accesses This kernel has uncoalesced global accesses resulting in a total of 6291456 excessive sectors (3% of the total 254972423 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.

Est. Speedup: 0.98%

Key Performance Indicators

L2 Theoretical Sectors Global Excessive

Location	Value	Value (%)
0x7e7df0274260 in dgemm_fp64	196,608	3
0x7e7df0274290 in dgemm_fp64	196,608	3
0x7e7df0274380 in dgemm_fp64	196,608	3
0x7e7df0274230 in dgemm_fp64	196,608	3
0x7e7df0274220 in dgemm_fp64	196,608	3

Uncoalesced Shared Accesses This kernel has uncoalesced shared accesses resulting in a total of 134217728 excessive wavefronts (38% of the total 352321536 wavefronts). Check the L1 Wavefronts Shared Excessive table for the primary source locations. The CUDA Best Practices Guide has an example on optimizing shared memory accesses.

Est. Speedup: 37.80%

Key Performance Indicators

L1 Wavefronts Shared Excessive

Location	Value	Value (%)
0x7e7df0273a50 in dgemm_fp64	4,194,304	3
0x7e7df0273960 in dgemm_fp64	4,194,304	3
0x7e7df0270510 in dgemm_fp64	4,194,304	3
0x7e7df0273850 in dgemm_fp64	4,194,304	3
0x7e7df0273800 in dgemm_fp64	4,194,304	3

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.