	Result	Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current	667 - poisson_single (32, 32, 16)x(8, 8, 16)	342.50 usecond	261,217	26	0 - NVIDIA A100-PCIE-40GB	762.66 cycle/usecond	8.0	[24124] poisson_solver

The report contains imported source files.

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 [%]	35.14	Duration [usecond]	342.50
Memory Throughput [%]	83.97	Elapsed Cycles [cycle]	261217
L1/TEX Cache Throughput [%]	75.35	SM Active Cycles [cycle]	258201.51
L2 Cache Throughput [%]	97.33	SM Frequency [cycle/usecond]	762.66
DRAM Throughput [%]	75.68	DRAM Frequency [cycle/nsecond]	1.21

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 L2 in the Memory Workload Analysis section.

Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 2:1. The kernel achieved 0% of this device's fp32 peak performance and 7% 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]	1.40	SM Busy [%]	35.55
Executed Ipc Active [inst/cycle]	1.42	Issue Slots Busy [%]	35.55
Issued Ipc Active [inst/cycle]	1.42		

Balanced ALU is the highest-utilized pipeline (20.6%) 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 [Tbyte/second]	1.17	Mem Busy [%]	83.97
L1/TEX Hit Rate [%]	51.76	Max Bandwidth [%]	75.68
L2 Hit Rate [%]	61.62	Mem Pipes Busy [%]	20.33
L2 Compression Success Rate [%]	0	L2 Compression Ratio	0

L1TEX Global Load Access Pattern The memory access pattern for global loads in L1TEX might not be optimal. On average, this kernel accesses 8.0 bytes per thread per memory request; but the address pattern, possibly caused by the stride between threads, results in 9.0 sectors per request, or 9.0*32 = 288.9 bytes of cache data transfers per request. The optimal thread address pattern for 8.0 byte accesses would result in 8.0*32 = 256.0 bytes of cache data transfers per request, to maximize L1TEX cache performance. Check the Source Counters section for uncoalesced global loads.

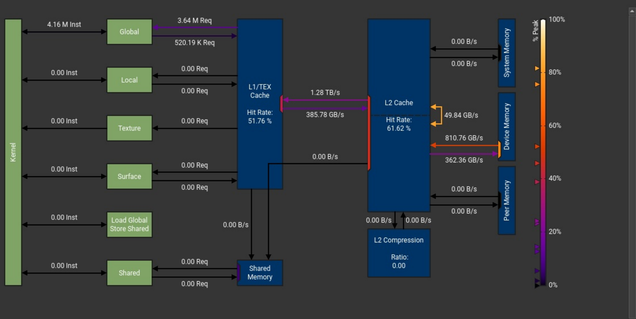
L2 Load Access Pattern The memory access pattern for loads from L1TEX to L2 is not optimal. The granularity of an L1TEX request to L2 is a 128 byte cache line. That is 4 consecutive 32-byte sectors per L2 request. However, this kernel only accesses an average of 1.6 sectors out of the possible 4 sectors 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.

L2 Store Access Pattern The memory access pattern for stores 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 2.0 sectors out of the possible 4 sectors per cache line. Check the Source Counters section for uncoalesced stores and try to minimize how many cache lines need to be accessed per memory request.

DRAM Excessive Read Sectors The memory access pattern for loads from device memory causes 10,471,408 sectors to be read from DRAM, which is 1.1x of the 9,395,421 sectors causing a miss in the L2 cache. The DRAM fetch granularity for read misses in L2 is 64 bytes, i.e. the lower or upper half of an L2 cache line. Try changing your access pattern to make use of both sectors returned by a DRAM read request for optimal usage of the DRAM throughput. For strided memory reads, avoid strides of 64 bytes or larger to avoid moving unused sectors from DRAM to L2.

Memory Chart

Show As: Throughput



Shared Memory

	Instructions	Requests	Wavefronts	% Peak	Bank Conflicts
Shared Load	0	0	0	0	0
Shared Load Matrix	0	0	0	0	0
Shared Store	0	0	0	0	0
Shared Store From Global Load	0	0	0	0	0
Shared Atomic	0	0	0	0	0
Other	-	-	1576960	5.59	0
Total	0	0	1576960	5.59	0

L1/TEX Cache

	Instructions	Requests	Wavefronts	% Peak	Sectors	Sectors/Req	Hit Rate	Bytes
Local Load	0	0	0	0	0	0	0	0
Global Load	3637248	3637248	-	-	32838136	9.03	58.19	1050820352
Global Load To Shared Store (access)	0	0	5118563	18.14	0	0		0
Global Load To Shared Store (bypass)	0	0	-	-	0	-		0
Surface Load	0	0	0	0	0	0		0
Texture Load	0	0	0	0	0	0		0
Global Store	520192	520192	533318	1.89	4129024	7.94	0.61	132128768
Local Store	0	0	0	0	0	0		0
Surface Store	0	0	0	0	0	0		0
Global Reduction	0	0	0	0	0	0		0
Surface Reduction	0	0	0	0	0	0		0
Global Atomic ALU	0	0	-	-	-	-		-
Global Atomic CAS	0	0	0	0	0	0		0
Surface Atomic ALU	0	0	-	-	-	-		-
Surface Atomic CAS	0	0	0	0	0	0		0
Loads	3637248	3637248	5118563	18.14	32838136	9.03	58.19	1050820352
Stores	520192	520192	533318	1.89	4129024	7.94	0.61	132128768
Atomics & Reductions	0	0	0	0	0	0		0

L2 Cache

	Requests	Sectors	Sectors/Req	% Peak	Hit Rate	Bytes	Throughput	Sector Misses to Device	Sec
L1/TEX Load	8828416	13727999	1.55	32.85	29.27	439295968	1282630944594.97	10445184	
L1/TEX Store	2064508	4129024	2.00	13.17	100	132128768	385781930299.92	0	
L1/TEX Atomic ALU	0	0	0	0	0	0	0	0	
L1/TEX Atomic CAS	0	0	0	0	0	0	0	0	
L1/TEX Reduction	0	0	0	0	0	0	0	0	
L1/TEX Total	10892457	17857054	1.64	42.73	44.65	571425728	1668415771279.08	10449726	
ECC Total	-	121150	-	0.07	-	3876800	11319256283.39	121150	
L2 Fabric Total	6444334	17070876	2.65	81.69	76.99	546268032	1594961786415.02	3925996	

L2 Cache Eviction Policies

	First	Hit Rate	Last	Hit Rate	Normal	Hit Rate	Normal Demote	Hit Rate
L1/TEX Load	0	0	0	0	14805714	29.17	0	0
L1/TEX Store	0	0	0	0	4129024	100	0	0
L1/TEX Atomic	0	0	0	0	0	0	-	-
L1/TEX Total	0	0	0	0	18933841	44.64	0	0
L2 Fabric Total	0	0	0	0	7900200	50.35	0	0
GPU Total	126779	4.42	0	0	26923524	46.15	0	0

Device Memory

	Sectors	% Peak	Bytes	Throughput
Load	8677616	52.31	277683712	810764832290.01
Store	3878352	23.38	124107264	362361207138.19
Total	12555968	75.68	401790976	1173126039428.20

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]	12.69	No Eligible [%]	64.40
Eligible Warps Per Scheduler [warp]	1.24	One or More Eligible [%]	35.60
Issued Warp Per Scheduler	0.36		

Issue Slot Utilization Every scheduler is capable of issuing one instruction per cycle, but for this kernel each scheduler only issues an instruction every 2.8 cycles. This might leave hardware resources underutilized and may lead to less optimal performance. Out of the maximum of 16 warps per scheduler, this kernel allocates an average of 12.69 active warps per scheduler, but only an average of 1.24 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]	35.63	Avg. Active Threads Per Warp	31.63
Warp Cycles Per Executed Instruction [cycle]	35.67	Avg. Not Predicated Off Threads Per Warp	31.18

long_scoreboard On average, each warp of this kernel spends 23.4 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 65.6% of the total average of 35.6 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]	39612416	Avg. Executed Instructions Per Scheduler [inst]	91695.41
Issued Instructions [inst]	39655867	Avg. Issued Instructions Per Scheduler [inst]	91795.99

FP32/64 Instructions This kernel executes 520192 fused and 3121152 non-fused FP64 instructions. By converting pairs of non-fused instructions to their higher-throughput equivalent, the achieved FP64 performance could be increased by up to 43% (relative to its current performance). Check the Source page to identify where this kernel executes FP64 instructions.

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	16384	Function Cache Configuration	CachePreferNone
Registers Per Thread [register/thread]	26	Static Shared Memory Per Block [byte/block]	0
Block Size	1024	Dynamic Shared Memory Per Block [byte/block]	0
Threads [thread]	16777216	Driver Shared Memory Per Block [kbyte/block]	1.02
Waves Per SM	75.85	Shared Memory Configuration Size [Kbyte]	8.19

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]	2
Theoretical Active Warps per SM [warp]	64	Block Limit Shared Mem [block]	8
Achieved Occupancy [%]	79.80	Block Limit Warps [block]	2
Achieved Active Warps Per SM [warp]	51.07	Block Limit SM [block]	32

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 (79.8%) 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. 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]	1044480	Branch Efficiency [%]	0
Branch Instructions Ratio [%]	0.03	Avg. Divergent Branches	0

Uncoalesced Global Accesses This kernel has uncoalesced global accesses resulting in a total of 4284224 excessive sectors (12% of the total 36967160 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 (%)
jacobi_cuda.cu:103 (0x7f4d7f452d00 in poisson_single)	2,047,748	48
jacobi_cuda.cu:102 (0x7f4d7f452ce6 in poisson_single)	2,047,748	48
jacobi_cuda.cu:106 (0x7f4d7f452d4d in poisson_single)	31,496	1
jacobi_cuda.cu:106 (0x7f4d7f452d10 in poisson_single)	31,496	1
jacobi_cuda.cu:101 (0x7f4d7f452cd0 in poisson_single)	31,496	1

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.