	Launch	Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current	1316 - cuda_parallel_launch_local_memory (13108, 1, 1)x(1, 160, 1)	18.52 second	14,158,869,643	79	0 - NVIDIA A100-PCIE-40GB	764.59 cycle/usecond	8.0	[1342261] raytrace-kokkos

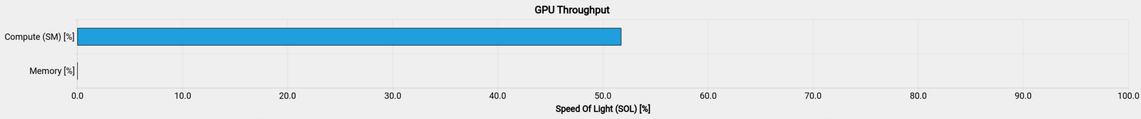
GPU Speed Of Light Throughput

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 [%]	51.73	Duration [second]	18.52
Memory Throughput [%]	0.00	Elapsed Cycles [cycle]	14,158,869,643
L1/TEX Cache Throughput [%]	0.00	SM Active Cycles [cycle]	14,087,915,685.33
L2 Cache Throughput [%]	0.00	SM Frequency [cycle/usecond]	764.59
DRAM Throughput [%]	0.00	DRAM Frequency [cycle/nsecond]	1.21

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 2:1. The kernel achieved 7% of this device's fp32 peak performance and close to 0% of its fp64 peak performance. See the Kernel Profiling Guide for mode details on roofline analysis.



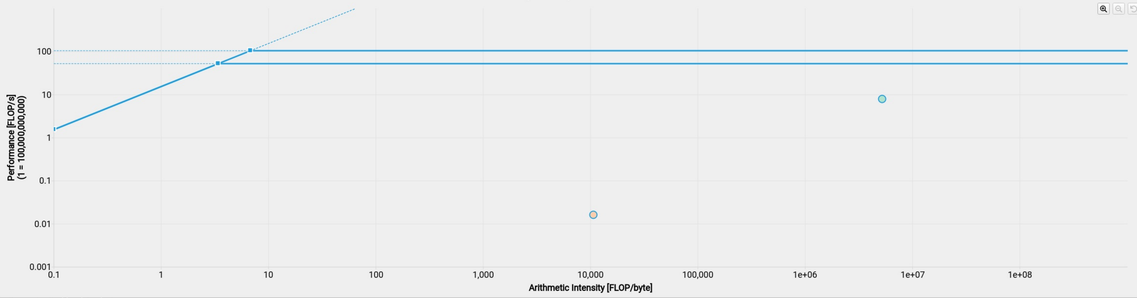
Compute Throughput Breakdown

SM: Pipe Fma Cycles Active [%]	51.73
SM: Issue Active [%]	41.09
SM: Inst Executed [%]	41.09
SM: Pipe Alu Cycles Active [%]	18.66
SM: Inst Executed Pipe Xu [%]	13.32
SM: Inst Executed Pipe Cbu Pred On Any [%]	7.94
SM: Pipe Shared Cycles Active [%]	2.22
SM: Pipe Tensor Cycles Active [%]	1.79
SM: Pipe Fp64 Cycles Active [%]	0.43
SM: Mio Pq Read Cycles Active [%]	0.01
SM: Mio Pq Write Cycles Active [%]	0.01
SM: Inst Executed Pipe Lsu [%]	0.00
SM: Inst Executed Pipe Adu [%]	0.00
SM: Mio Inst Issued [%]	0.00
SM: Mio2rf Writeback Active [%]	0.00
SM: Inst Executed Pipe Uniform [%]	0.00
IDC: Request Cycles Active [%]	0
SM: Inst Executed Pipe Tex [%]	0
SM: Inst Executed Pipe Ipa [%]	0
SM: Inst Executed Pipe Fp16 [%]	0

Memory Throughput Breakdown

L2: Xbar2lts Cycles Active [%]	0.00
L2: T Tag Requests [%]	0.00
L1: M L1tex2xbar Req Cycles Active [%]	0.00
L2: T Sectors [%]	0.00
L1: Data Pipe Lsu Wavefronts [%]	0.00
L2: D Sectors [%]	0.00
L2: Lts2xbar Cycles Active [%]	0.00
L1: Lsuin Requests [%]	0.00
L1: Data Bank Reads [%]	0.00
DRAM: Cycles Active [%]	0.00
DRAM: Dram Sectors [%]	0.00
L2: D Sectors Fill Device [%]	0.00
L1: Data Bank Writes [%]	0.00
L1: Lsu Writeback Active [%]	0.00
L2: D Sectors Fill Sysmem [%]	0.00
L1: Texin Sm2tex Req Cycles Active [%]	0.00
L1: F Wavefronts [%]	0.00
L2: D Atomic Input Cycles Active [%]	0
L1: Tex Writeback Active [%]	0
L1: M Xbar2l1tex Read Sectors [%]	0
L1: Data Pipe Tex Wavefronts [%]	0

Floating Point Operations Roofline



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]	1.64	SM Busy [%]	51.99
Executed Ipc Active [inst/cycle]	1.65	Issue Slots Busy [%]	41.30
Issued Ipc Active [inst/cycle]	1.65		

Balanced No pipeline is over-utilized.

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 [Kbyte/second]	148.15	Mem Busy [%]	0.00
L1/TEX Hit Rate [%]	70.85	Max Bandwidth [%]	0.00
L2 Hit Rate [%]	100.09	Mem Pipes Busy [%]	0.01
L2 Compression Success Rate [%]	0	L2 Compression Ratio	0

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]	4.85	No Eligible [%]	58.69
Eligible Warps Per Scheduler [warp]	0.71	One or More Eligible [%]	41.31
Issued Warp Per Scheduler	0.41		

Issue Slot Utilization Every scheduler is capable of issuing one instruction per cycle, but for this kernel each scheduler only issues an instruction every 2.4 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 4.85 active warps per scheduler, but only an average of 0.71 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 5.00 theoretical warps per scheduler this kernel can issue according to its occupancy are below the hardware maximum of 16. 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]	11.73	Avg. Active Threads Per Warp	6.72
Warp Cycles Per Executed Instruction [cycle]	11.74	Avg. Not Predicated Off Threads Per Warp	6.51

no_instruction On average, each warp of this kernel spends 7.2 cycles being stalled due to not having the next instruction fetched yet. This represents about 61.3% of the total average of 11.7 cycles between issuing two instructions. A high number of warps not having an instruction fetched is typical for very short kernels with less than one full wave of work in the grid. Excessively jumping across large blocks of assembly code can also lead to more warps stalled for this reason, if this causes misses in the instruction cache.

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.

Thread Divergence Instructions are executed in warps, which are groups of 32 threads. Optimal instruction throughput is achieved if all 32 threads of a warp execute the same instruction. The chosen launch configuration, early thread completion, and divergent flow control can significantly lower the number of active threads in a warp per cycle. This kernel achieves an average of 6.7 threads being active per cycle. This is further reduced to 6.5 threads per warp due to predication. The compiler may use predication to avoid an actual branch. Instead, all instructions are scheduled, but a per-thread condition code or predicate controls which threads execute the instructions. Try to avoid different execution paths within a warp when possible. In addition, ensure your kernel makes use of Independent Thread Scheduling, which allows a warp to reconverge after a data-dependent conditional block by explicitly calling __syncwarp().

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]	2,513,145,840,212	Avg. Executed Instructions Per Scheduler [inst]	5,817,467,222.71
Issued Instructions [inst]	2,513,219,944,089	Avg. Issued Instructions Per Scheduler [inst]	5,817,638,759.47

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	13,108	Registers Per Thread [register/thread]	79
Block Size	160	Static Shared Memory Per Block [byte/block]	0
Threads [thread]	2,097,280	Dynamic Shared Memory Per Block [byte/block]	0
Waves Per SM	30.34	Driver Shared Memory Per Block [Kbyte/block]	1.02
Function Cache Configuration	cudaFuncCachePreferL1	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 [%]	31.25	Block Limit Registers [block]	4
Theoretical Active Warps per SM [warp]	20	Block Limit Shared Mem [block]	164
Achieved Occupancy [%]	30.27	Block Limit Warps [block]	12
Achieved Active Warps Per SM [warp]	19.37	Block Limit SM [block]	32

Occupancy Limiters This kernel's theoretical occupancy (31.2%) is limited by the number of required registers 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]	137,250,030,406	Branch Efficiency [%]	97.78
Branch Instructions Ratio [%]	0.05	Avg. Divergent Branches	3,883,997.34

Uncoalesced Global Accesses Uncoalesced global access, expected 65536 sectors, got 2097152 (32.00x) at PC 0x14f5c2fcd970 at /vast/home/pat/Projects/kitsune/kitsune/experiments/raytracer/raytrace-kokkos.cpp:194

Uncoalesced Global Accesses Uncoalesced global access, expected 65536 sectors, got 2097152 (32.00x) at PC 0x14f5c2fcd9b0 at /vast/home/pat/Projects/kitsune/kitsune/experiments/raytracer/raytrace-kokkos.cpp:195

Uncoalesced Global Accesses Uncoalesced global access, expected 65536 sectors, got 2097152 (32.00x) at PC 0x14f5c2fcd9e0 at /vast/home/pat/Projects/kitsune/kitsune/experiments/raytracer/raytrace-kokkos.cpp:196

Uncoalesced Global Accesses See the CUDA Programming Guide for reducing uncoalesced device memory accesses.