Page: Details | Launch: 4 - 731 - _ZN6Kokkos4Imp | Add Baseline | Apply Rules | Occupancy Calculator | Save as PDF

	Launch	Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current	731 - _ZN6Kokkos4ImplL33cuda_parallel_launch_local_memoryINS0_11ParallelForIZ4mainEUlRKiE_NS_11RangePolicyIJNS_4CudaEEEES7_EEEEvT...	15.15 second	11,585,407,796	82	0 - NVIDIA A100-PCIE-40GB	764.57 cycle/usecond	8.0	[1343071] raytrace-clang

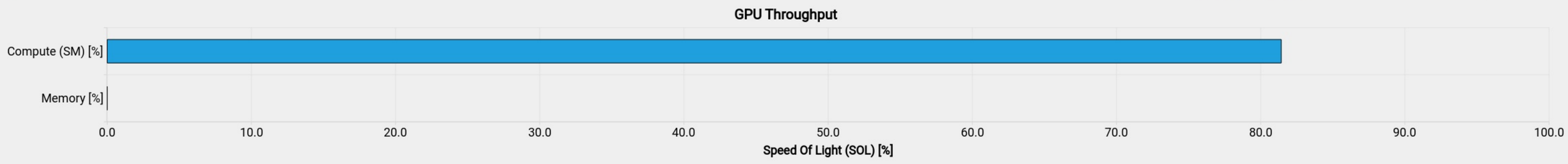
GPU Speed Of Light Throughput

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 [%]	81.43	Duration [second]	15.15
Memory Throughput [%]	0.00	Elapsed Cycles [cycle]	11,585,407,796
L1/TEX Cache Throughput [%]	0.00	SM Active Cycles [cycle]	11,541,652,897.51
L2 Cache Throughput [%]	0.00	SM Frequency [cycle/usecond]	764.57
DRAM Throughput [%]	0.00	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 workloads in the Compute Workload Analysis section.

Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 2:1. The kernel achieved 13% 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.

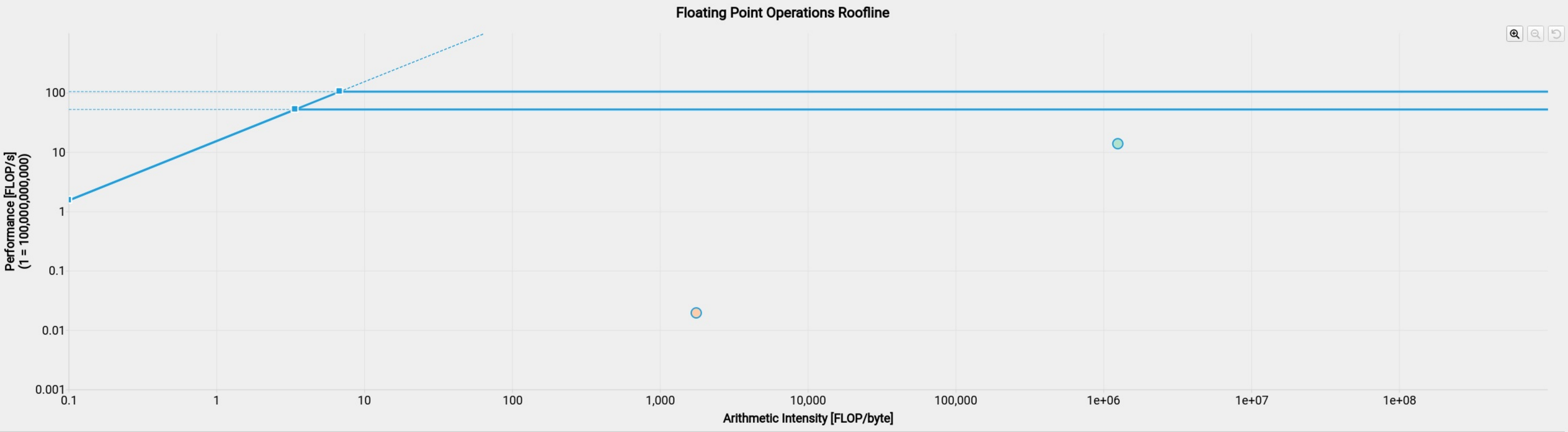


Compute Throughput Breakdown

SM: Pipe Fma Cycles Active [%]	81.43
SM: Issue Active [%]	65.67
SM: Inst Executed [%]	65.67
SM: Pipe Alu Cycles Active [%]	27.76
SM: Inst Executed Pipe Xu [%]	26.75
SM: Inst Executed Pipe Adu [%]	14.89
IDC: Request Cycles Active [%]	14.89
SM: Inst Executed Pipe Cbu Pred On Any [%]	11.41
SM: Mio Pq Read Cycles Active [%]	7.46
SM: Mio Pq Write Cycles Active [%]	7.46
SM: Mio Inst Issued [%]	3.72
SM: Mio2rf Writeback Active [%]	3.72
SM: Pipe Shared Cycles Active [%]	0.97
SM: Pipe Fp64 Cycles Active [%]	0.52
SM: Pipe Tensor Cycles Active [%]	0.45
SM: Inst Executed Pipe Lsu [%]	0.00
SM: Inst Executed Pipe Uniform [%]	0.00
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
L1: M L1tex2xbar Req Cycles Active [%]	0.00
L2: T Tag Requests [%]	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
L1: Data Bank Writes [%]	0.00
L1: Lsu Writeback Active [%]	0.00
DRAM: Cycles Active [%]	0.00
DRAM: Dram Sectors [%]	0.00
L1: M Xbar2l1tex Read Sectors [%]	0.00
L2: D Sectors Fill Device [%]	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: Data Pipe Tex Wavefronts [%]	0



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]	2.63	SM Busy [%]	81.74
Executed Ipc Active [inst/cycle]	2.64	Issue Slots Busy [%]	65.92
Issued Ipc Active [inst/cycle]	2.64		

Very High Utilization FMA is the highest-utilized pipeline (81.7%). It executes 32-bit floating point (FADD, FMUL, FMAD, ...) and integer (IMUL, IMAD) operations. The pipeline is over-utilized and likely a performance bottleneck. See the Kernel Profiling Guide for the workloads handled by each pipeline.

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 [Mbyte/second]	1.09	Mem Busy [%]	0.00
L1/TEX Hit Rate [%]	64.51	Max Bandwidth [%]	0.00
L2 Hit Rate [%]	99.80	Mem Pipes Busy [%]	14.89
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]	4.70	No Eligible [%]	34.05
Eligible Warps Per Scheduler [warp]	1.58	One or More Eligible [%]	65.95
Issued Warp Per Scheduler	0.66		

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]	7.13	Avg. Active Threads Per Warp	6.64
Warp Cycles Per Executed Instruction [cycle]	7.13	Avg. Not Predicated Off Threads Per Warp	6.51

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 6.6 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]	3,286,667,186,026	Avg. Executed Instructions Per Scheduler [inst]	7,608,025,893.58
Issued Instructions [inst]	3,286,761,685,604	Avg. Issued Instructions Per Scheduler [inst]	7,608,244,642.60

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	16,384	Registers Per Thread [register/thread]	82
Block Size	128	Static Shared Memory Per Block [byte/block]	0
Threads [thread]	2,097,152	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]	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 [%]	31.25	Block Limit Registers [block]	5
Theoretical Active Warps per SM [warp]	20	Block Limit Shared Mem [block]	164
Achieved Occupancy [%]	29.38	Block Limit Warps [block]	16
Achieved Active Warps Per SM [warp]	18.80	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]	205,327,703,225	Branch Efficiency [%]	98.72
Branch Instructions Ratio [%]	0.06	Avg. Divergent Branches	3,932,338.44

Uncoalesced Global Accesses Uncoalesced global access, expected 65536 sectors, got 2097152 (32.00x) at PC 0x149e24fcad20

Uncoalesced Global Accesses Uncoalesced global access, expected 65536 sectors, got 2097152 (32.00x) at PC 0x149e24fcabd0

Uncoalesced Global Accesses Uncoalesced global access, expected 65536 sectors, got 2097152 (32.00x) at PC 0x149e24fcac70

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