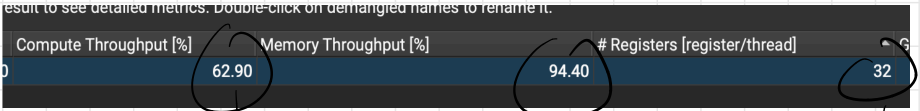


esult to see detailed metrics. Double-click on demangled names to rename it.

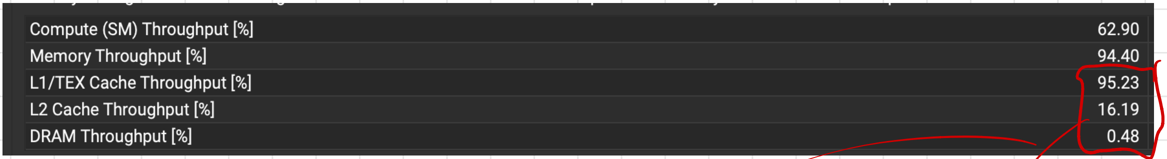
Compute Throughput [%]	Memory Throughput [%]	# Registers [register/thread]	G
62.90	94.40	32	

much more stuff to do in matmul than in vector add (higher AI)

really more usage of L1/L2 as opposed to just HBM w/ (higher data reuse)

also use some strided access (hard)

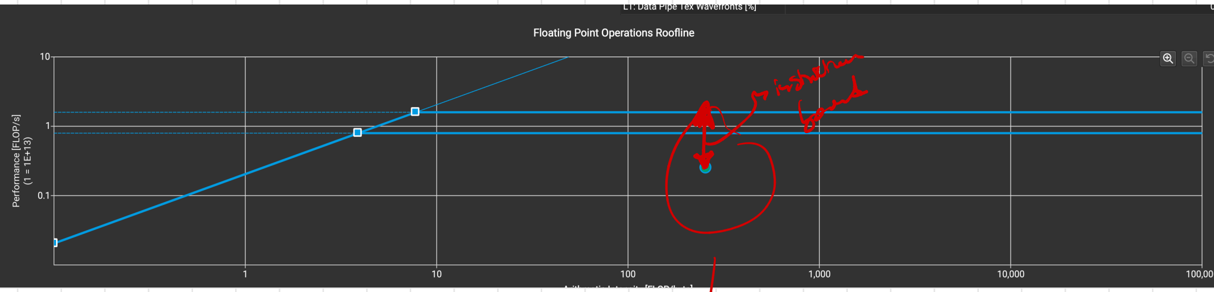
2x from vector add



Compute (SM) Throughput [%]	62.90
Memory Throughput [%]	94.40
L1/TEX Cache Throughput [%]	95.23
L2 Cache Throughput [%]	16.19
DRAM Throughput [%]	0.48

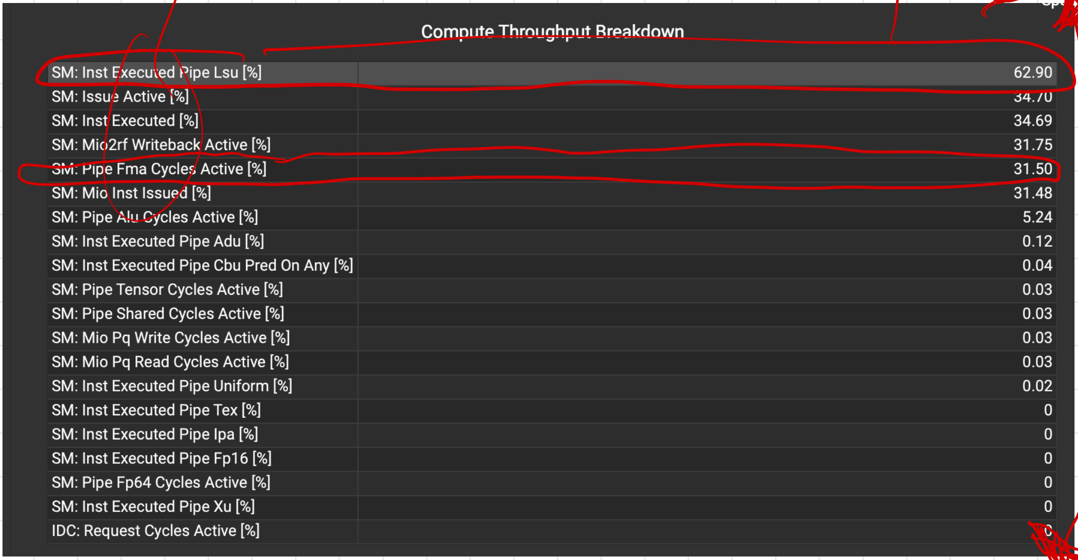
* L1-bound *

Small working set → lives in L1 → Dram / L2 low usage



Q → WAY TOO MANY L/S ops → compute bound

→ Instruction bound

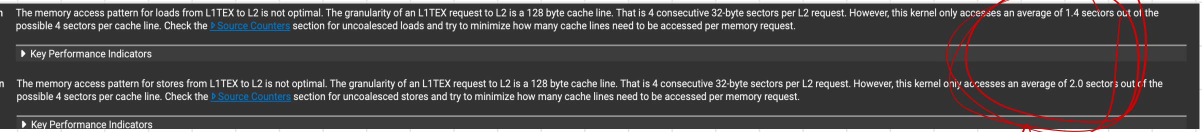


Compute Throughput Breakdown

SM: Inst Executed Pipe Lsu [%]	62.90
SM: Issue Active [%]	34.70
SM: Inst Executed [%]	34.69
SM: Mio2rf Writeback Active [%]	31.75
SM: Pipe Fma Cycles Active [%]	31.50
SM: Mio Inst Issued [%]	31.48
SM: Pipe Alu Cycles Active [%]	5.24
SM: Inst Executed Pipe Adu [%]	0.12
SM: Inst Executed Pipe Cbu Pred On Any [%]	0.04
SM: Pipe Tensor Cycles Active [%]	0.03
SM: Pipe Shared Cycles Active [%]	0.03
SM: Mio Pq Write Cycles Active [%]	0.03
SM: Mio Pq Read Cycles Active [%]	0.03
SM: Inst Executed Pipe Uniform [%]	0.02
SM: Inst Executed Pipe Tex [%]	0
SM: Inst Executed Pipe Ipa [%]	0
SM: Inst Executed Pipe Fp16 [%]	0
SM: Pipe Fp64 Cycles Active [%]	0
SM: Inst Executed Pipe Xu [%]	0
IDC: Request Cycles Active [%]	0

63% of time SM doing L/s to L1

32% of time it is doing math,



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 1.4 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.

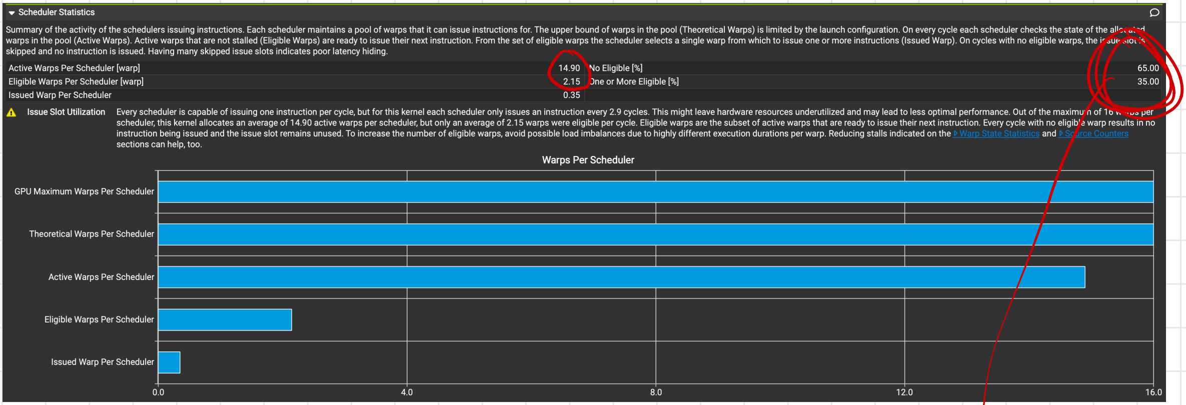
▸ Key Performance Indicators

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.

▸ Key Performance Indicators

stride access is killing us

so much waste



▾ 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]	14.90	No Eligible [%]	65.00
Eligible Warps Per Scheduler [warp]	2.15	One or More Eligible [%]	35.00
Issued Warp Per Scheduler	0.35		

⚠ **Issue Slot Utilization** Every scheduler is capable of issuing one instruction per cycle, but for this kernel each scheduler only issues an instruction every 2.9 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 14.90 active warps per scheduler, but only an average of 2.15 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.

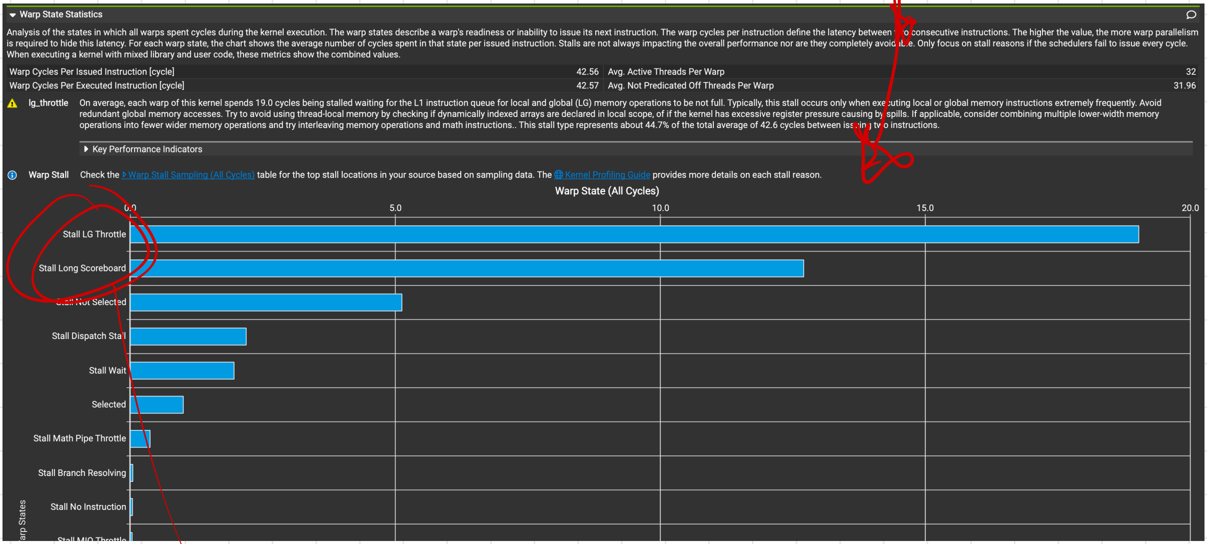
Warps Per Scheduler

GPU Maximum Warps Per Scheduler
Theoretical Warps Per Scheduler
Active Warps Per Scheduler
Eligible Warps Per Scheduler
Issued Warp Per Scheduler
0.0
4.0
8.0
12.0
16.0

very low eligibility

(waiting for uncoalesced)

STALL 45



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]	42.56	Avg. Active Threads Per Warp	32
Warp Cycles Per Executed Instruction [cycle]	42.57	Avg. Not Predicated Off Threads Per Warp	31.96

⚠ lg_throttle On average, each warp of this kernel spends 19.0 cycles being stalled waiting for the L1 instruction queue for local and global (LG) memory operations to be not full. Typically, this stall occurs only when executing local or global memory instructions extremely frequently. Avoid redundant global memory accesses. Try to avoid using thread-local memory by checking if dynamically indexed arrays are declared in local scope, of if the kernel has excessive register pressure causing by spills. If applicable, consider combining multiple lower-width memory operations into fewer wider memory operations and try interleaving memory operations and math instructions.. This stall type represents about 44.7% of the total average of 42.6 cycles between issuing two instructions.

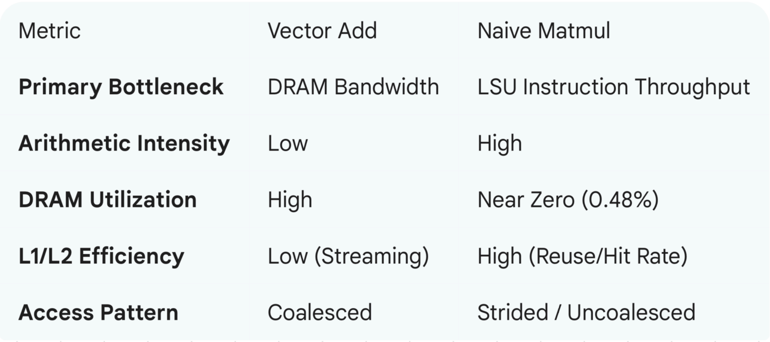
▸ Key Performance Indicators

ⓘ 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.

again stalled on LD/ST

Wavefronts at 94.4%.

overhead



Metric	Vector Add	Naive Matmul
Primary Bottleneck	DRAM Bandwidth	LSU Instruction Throughput
Arithmetic Intensity	Low	High
DRAM Utilization	High	Near Zero (0.48%)
L1/L2 Efficiency	Low (Streaming)	High (Reuse/Hit Rate)
Access Pattern	Coalesced	Strided / Uncoalesced