	Result	Size	Time	Cycles	GPU	SM Frequency	Process	Attributes
Current	577 - noiseReduction_kernel	(120, 68, 1)x(16, 16, 1)	226.69 us	326,112	0 - NVIDIA GeForce GTX 1660 Ti	1.43 Ghz	[15036] image	

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

Compare | Tools | View | Export

This table shows all results in the report. Use the column headers to sort the results in this report. Double-click a result to see detailed metrics. Double-click on demangled names to rename it.

ID	Estimated Speedup [%]	Function Name	Demangled Name	Duration [ms] (4.64 ms)	Runtime Improvement [ms] (2.73 ms)	Compute Throughput [%]	Memory Throughput [%]	# Registers [regi
0	93.84	init_cos_sin_table_...	init_cos_sin_table_...	0.00	0.00	0.33	2.37	
1	85.64	image_RGB2BW_ke...	image_RGB2BW_ke...	0.24	0.20	88.13	12.77	
2	62.51	noiseReduction_ker...	noiseReduction_ker...	0.23	0.14	46.62	48.64	

The following performance optimization opportunities were discovered for this result. Follow the rule links to see more context on the Details page.
Note: *Speedup estimates provide upper bounds for the optimization potential of a kernel assuming its overall algorithmic structure is kept unchanged.*

L1TEX Global Load Access Pattern
Est. Speedup: 62.51%
The memory access pattern for global loads from L1TEX might not be optimal. On average, only 11.5 of the 32 bytes transmitted per sector are utilized by each thread. This could possibly be caused by a stride between threads. Check the Source Counters section for uncoalesced global loads.

Uncoalesced Global Accesses
Est. Speedup: 58.57%
This kernel has uncoalesced global accesses resulting in a total of 2883680 excessive sectors (61% of the total 4755920 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.

Lg Throttle Stalls
Est. Speedup: 51.36%
On average, each warp of this workload spends 14.6 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 57.0% of the total average of 25.5 cycles between issuing two instructions.