	Result	Size	Time	Cycles	GPU	SM Frequency	Process	Attributes
Current	589 - hough_kernel	(120, 68, 1)x(16, 16, 1)	891.42 us	1,291,471	0 - NVIDIA GeForce GTX 1660 Ti	1.44 Ghz	[11724] 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] (6.45 ms)	Runtime Improvement [ms] (4.27 ms)	Compute Throughput [%]	Memory Throughput [%]	# Reg
5	48.34	thresholding_kernel	thresholding_kernel..	0.09	0.04	32.00	51.66	
6	64.29	hough_kernel	hough_kernel..char..	0.89	0.57	83.52	17.84	
7	60.69	getlines_kernel	getlines_kernel..un...	0.02	0.01	17.68	60.94	

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.*

Thread Divergence
Est. Speedup: 64.29%

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 workload achieves an average of 7.6 threads being active per cycle. This is further reduced to 7.4 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.

FP64 Non-Fused Instructions
Est. Speedup: 43.34%

This kernel executes 0 fused and 529020 non-fused FP64 instructions. By converting pairs of non-fused instructions to their fused, higher-throughput equivalent, the achieved FP64 performance could be increased by up to 50% (relative to its current performance). Check the Source page to identify where this kernel executes FP64 instructions.

Uncoalesced Global Accesses
Est. Speedup: 40.17%

This kernel has uncoalesced global accesses resulting in a total of 1230354 excessive sectors (42% of the total 2959749 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.