	Result	Size	Time	Cycles	GPU	SM Frequency	Process	Attributes
Current	572 - runJacobiCUDA_kernel1	(32, 128, 1)x(32, 8, 1)	14.30 us	32,943	0 - NVIDIA GeForce RTX 5060 Ti	2.29 Ghz	[337018] jacobi2D_baseline_1024.exe	

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 [us] (2,780.22 us)	Runtime Improvement [us] (873.51 us)	Compute Throughput [%]	Memory Throughput [%]	# Registers [register/thread	Grid Size	Block Size [block]
0	33.28	**runJacobiCUDA_kernel1**	**runJacobiCUDA_kernel1(int, float *, float *)**	14.30	4.76	44.43	66.72	18	32, 128, ...	32, 8,
1	29.47	runJacobiCUDA_kernel2	runJacobiCUDA_kernel2(int, float *, float *)	13.50	3.98	35.04	70.53	16	32, 128, ...	32, 8,
2	33.38	runJacobiCUDA_kernel1	runJacobiCUDA_kernel1(int, float *, float *)	14.30	4.77	44.41	66.62	18	32, 128, ...	32, 8,
3	28.68	runJacobiCUDA_kernel2	runJacobiCUDA_kernel2(int, float *, float *)	13.38	3.84	35.44	71.32	16	32, 128, ...	32, 8,
4	32.85	runJacobiCUDA_kernel1	runJacobiCUDA_kernel1(int, float *, float *)	14.21	4.67	44.73	67.15	18	32, 128, ...	32, 8,
5	28.48	runJacobiCUDA_kernel2	runJacobiCUDA_kernel2(int, float *, float *)	13.31	3.79	35.56	71.52	16	32, 128, ...	32, 8,
6	33.37	runJacobiCUDA_kernel1	runJacobiCUDA_kernel1(int, float *, float *)	14.34	4.78	44.36	66.63	18	32, 128, ...	32, 8,
7	28.78	runJacobiCUDA_kernel2	runJacobiCUDA_kernel2(int, float *, float *)	13.38	3.85	35.39	71.22	16	32, 128, ...	32, 8,
8	33.28	runJacobiCUDA_kernel1	runJacobiCUDA_kernel1(int, float *, float *)	14.30	4.76	44.53	66.72	18	32, 128, ...	32, 8,
9	28.88	runJacobiCUDA_kernel2	runJacobiCUDA_kernel2(int, float *, float *)	13.41	3.87	35.35	71.12	16	32, 128, ...	32, 8,
10	33.02	runJacobiCUDA_kernel1	runJacobiCUDA_kernel1(int, float *, float *)	14.27	4.71	44.59	66.98	18	32, 128, ...	32, 8,
11	29.76	runJacobiCUDA_kernel2	runJacobiCUDA_kernel2(int, float *, float *)	13.54	4.03	34.94	70.24	16	32, 128, ...	32, 8,
12	33.19	runJacobiCUDA_kernel1	runJacobiCUDA_kernel1(int, float *, float *)	14.27	4.74	44.58	66.81	18	32, 128, ...	32, 8,
13	30.62	runJacobiCUDA_kernel2	runJacobiCUDA_kernel2(int, float *, float *)	13.73	4.20	34.42	69.38	16	32, 128, ...	32, 8,
14	34.06	runJacobiCUDA_kernel1	runJacobiCUDA_kernel1(int, float *, float *)	14.46	4.93	43.86	65.94	18	32, 128, ...	32, 8,
15	28.98	runJacobiCUDA_kernel2	runJacobiCUDA_kernel2(int, float *, float *)	13.41	3.89	35.25	71.02	16	32, 128, ...	32, 8,
16	32.76	runJacobiCUDA_kernel1	runJacobiCUDA_kernel1(int, float *, float *)	14.21	4.66	44.86	67.24	18	32, 128, ...	32, 8,
17	28.98	runJacobiCUDA_kernel2	runJacobiCUDA_kernel2(int, float *, float *)	13.41	3.89	35.31	71.02	16	32, 128, ...	32, 8,
18	34.48	runJacobiCUDA_kernel1	runJacobiCUDA_kernel1(int, float *, float *)	14.53	5.01	43.67	65.52	18	32, 128, ...	32, 8,
19	28.78	runJacobiCUDA_kernel2	runJacobiCUDA_kernel2(int, float *, float *)	13.38	3.85	35.36	71.22	16	32, 128, ...	32, 8,
20	33.63	runJacobiCUDA_kernel1	runJacobiCUDA_kernel1(int, float *, float *)	14.37	4.83	44.23	66.37	18	32, 128, ...	32, 8,
21	28.98	runJacobiCUDA_kernel2	runJacobiCUDA_kernel2(int, float *, float *)	13.41	3.89	35.38	71.02	16	32, 128	32, 8

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.*

Long Scoreboard Stalls
Est. Speedup: 33.28%

On average, each warp of this workload spends 24.6 cycles being stalled waiting for a scoreboard dependency on a L1TEX (local, global, surface, texture) operation. Find the instruction producing the data being waited upon to identify the culprit. To reduce the number of cycles waiting on L1TEX data accesses verify the memory access patterns are optimal for the target architecture, attempt to increase cache hit rates by increasing data locality (coalescing), or by changing the cache configuration. Consider moving frequently used data to shared memory. This stall type represents about 74.3% of the total average of 33.1 cycles between issuing two instructions.

▸ Key Performance Indicators

Achieved Occupancy
Est. Speedup: 26.43%

The difference between calculated theoretical (100.0%) and measured achieved occupancy (73.6%) can be the result of warp scheduling overheads or workload imbalances during the kernel execution. Load imbalances can occur between warps within a block as well as across blocks of the same kernel. See the CUDA Best Practices Guide for more details on optimizing occupancy.

▸ Key Performance Indicators

Uncoalesced Global Accesses

This kernel has uncoalesced global accesses resulting in a total of 63364 excessive sectors (7% of the total 848260 sectors). Check the L2 Theoretical Sectors Global Excessive table for the primary source locations. The CUDA Programming Guide has additional information on