

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] (1,444.45 us)	Runtime Improvement [us] (490.14 us)	Compute Throughput [%]	Memory Throughput [%]	# Registers [register/thread]	Grid Size	Block Size [block]
0	34.64	jacobi_kernel	jacobi_kernel(int, fl...	14.59	5.05	43.67	65.36	20	32, 128, ...	32, 8, ...
1	33.63	jacobi_kernel	jacobi_kernel(int, fl...	14.37 (-1.54%)	4.83	44.40 (+1.66%)	66.37 (+1.55%)	20 (+0.00%)	32, 128, ...	32, 8, ...
2	33.63	jacobi_kernel	jacobi_kernel(int, fl...	14.34 (-1.75%)	4.82	44.45 (+1.79%)	66.37 (+1.55%)	20 (+0.00%)	32, 128, ...	32, 8, ...
3	33.82	jacobi_kernel	jacobi_kernel(int, fl...	14.40 (-1.32%)	4.87	44.21 (+1.24%)	66.18 (+1.26%)	20 (+0.00%)	32, 128, ...	32, 8, ...
4	33.89	jacobi_kernel	jacobi_kernel(int, fl...	14.43 (-1.10%)	4.89	44.31 (+1.47%)	66.11 (+1.16%)	20 (+0.00%)	32, 128, ...	32, 8, ...
5	33.52	jacobi_kernel	jacobi_kernel(int, fl...	14.37 (-1.54%)	4.82	44.43 (+1.73%)	66.48 (+1.72%)	20 (+0.00%)	32, 128, ...	32, 8, ...
6	32.84	jacobi_kernel	jacobi_kernel(int, fl...	14.24 (-2.41%)	4.68	44.90 (+2.82%)	67.16 (+2.75%)	20 (+0.00%)	32, 128, ...	32, 8, ...
7	33.37	jacobi_kernel	jacobi_kernel(int, fl...	14.34 (-1.75%)	4.78	44.50 (+1.90%)	66.63 (+1.94%)	20 (+0.00%)	32, 128, ...	32, 8, ...
8	32.94	jacobi_kernel	jacobi_kernel(int, fl...	14.21 (-2.63%)	4.68	44.92 (+2.85%)	67.06 (+2.61%)	20 (+0.00%)	32, 128, ...	32, 8, ...
9	33.80	jacobi_kernel	jacobi_kernel(int, fl...	14.40 (-1.32%)	4.87	44.31 (+1.45%)	66.20 (+1.29%)	20 (+0.00%)	32, 128, ...	32, 8, ...
10	32.94	jacobi_kernel	jacobi_kernel(int, fl...	14.24 (-2.41%)	4.69	44.87 (+2.74%)	67.06 (+2.61%)	20 (+0.00%)	32, 128, ...	32, 8, ...
11	33.45	jacobi_kernel	jacobi_kernel(int, fl...	14.37 (-1.54%)	4.81	44.44 (+1.76%)	66.55 (+1.82%)	20 (+0.00%)	32, 128, ...	32, 8, ...
12	33.46	jacobi_kernel	jacobi_kernel(int, fl...	14.34 (-1.75%)	4.80	44.51 (+1.93%)	66.54 (+1.81%)	20 (+0.00%)	32, 128, ...	32, 8, ...
13	33.63	jacobi_kernel	jacobi_kernel(int, fl...	14.37 (-1.54%)	4.83	44.33 (+1.51%)	66.37 (+1.55%)	20 (+0.00%)	32, 128, ...	32, 8, ...
14	33.45	jacobi_kernel	jacobi_kernel(int, fl...	14.34 (-1.75%)	4.80	44.46 (+1.82%)	66.55 (+1.82%)	20 (+0.00%)	32, 128, ...	32, 8, ...
15	34.31	jacobi_kernel	jacobi_kernel(int, fl...	14.53 (-0.44%)	4.98	43.91 (+0.55%)	65.69 (+0.51%)	20 (+0.00%)	32, 128, ...	32, 8, ...
16	33.71	jacobi_kernel	jacobi_kernel(int, fl...	14.40 (-1.32%)	4.85	44.28 (+1.38%)	66.29 (+1.42%)	20 (+0.00%)	32, 128, ...	32, 8, ...
17	34.73	jacobi_kernel	jacobi_kernel(int, fl...	14.59 (+0.00%)	5.07	43.60 (-0.17%)	65.27 (-0.13%)	20 (+0.00%)	32, 128, ...	32, 8, ...
18	33.37	jacobi_kernel	jacobi_kernel(int, fl...	14.34 (-1.75%)	4.78	44.50 (+1.89%)	66.63 (+1.94%)	20 (+0.00%)	32, 128, ...	32, 8, ...
19	34.22	jacobi_kernel	jacobi_kernel(int, fl...	14.53 (-0.44%)	4.97	43.95 (+0.63%)	65.78 (+0.64%)	20 (+0.00%)	32, 128, ...	32, 8, ...
20	33.37	jacobi_kernel	jacobi_kernel(int, fl...	14.34 (-1.75%)	4.78	44.50 (+1.90%)	66.63 (+1.95%)	20 (+0.00%)	32, 128, ...	32, 8, ...
21	34.23	jacobi_kernel	jacobi_kernel(int, fl...	14.50 (-0.66%)	4.96	44.00 (+0.75%)	65.77 (+0.64%)	20 (+0.00%)	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: 34.64%

On average, each warp of this workload spends 19.9 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 71.4% of the total average of 27.9 cycles between issuing two instructions.

► Key Performance Indicators

Achieved Occupancy

Est. Speedup: 26.40%

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