Analysis Report

correct(configurazione, int*, int)

Duration	152.546 µs
Grid Size	[34,1,1]
Block Size	[512,1,1]
Registers/Thread	30
Shared Memory/Block	0 B
Shared Memory Requested	64 KiB
Shared Memory Executed	64 KiB
Shared Memory Bank Size	4 B

[0] GeForce 840M

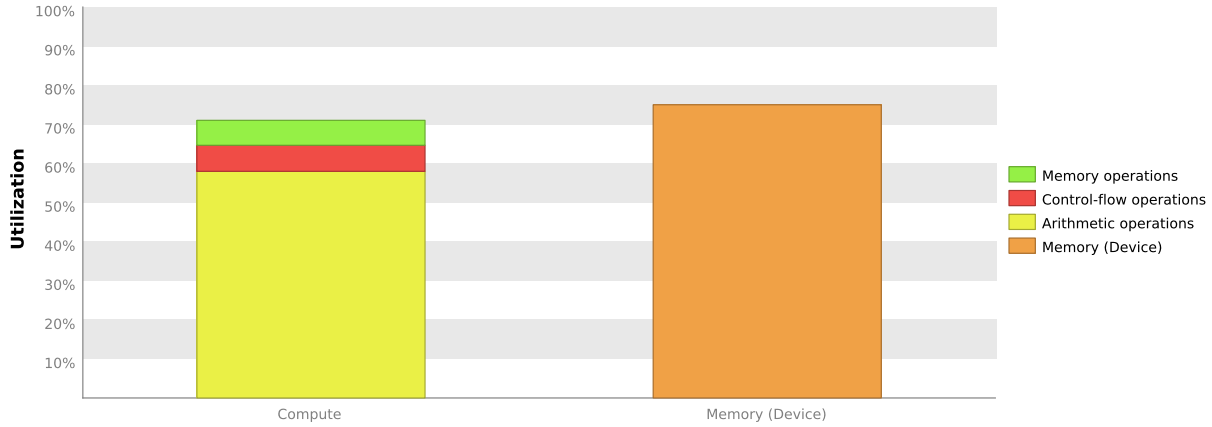
GPU UUID	GPU-4a39874c-8303-b7e3-9758-a265118f0297
Compute Capability	5.0
Max. Threads per Block	1024
Max. Shared Memory per Block	48 KiB
Max. Registers per Block	65536
Max. Grid Dimensions	[2147483647, 65535, 65535]
Max. Block Dimensions	[1024, 1024, 64]
Max. Warps per Multiprocessor	64
Max. Blocks per Multiprocessor	32
Single Precision FLOP/s	863.232 GigaFLOP/s
Double Precision FLOP/s	26.976 GigaFLOP/s
Number of Multiprocessors	3
Multiprocessor Clock Rate	1.124 GHz
Concurrent Kernel	true
Max IPC	6
Threads per Warp	32
Global Memory Bandwidth	14.4 GB/s
Global Memory Size	2 GiB
Constant Memory Size	64 KiB
L2 Cache Size	1 MiB
Memcpy Engines	1
PCIe Generation	2
PCIe Link Rate	5 Gbit/s
PCIe Link Width	4

1. Compute, Bandwidth, or Latency Bound

The first step in analyzing an individual kernel is to determine if the performance of the kernel is bounded by computation, memory bandwidth, or instruction/memory latency. The results below indicate that the performance of kernel "correct" is most likely limited by memory bandwidth. You should first examine the information in the "Memory Bandwidth" section to determine how it is limiting performance.

1.1. Kernel Performance Is Bound By Memory Bandwidth

For device "GeForce 840M" the kernel's compute utilization is significantly lower than its memory utilization. These utilization levels indicate that the performance of the kernel is most likely being limited by the memory system. For this kernel the limiting factor in the memory system is the bandwidth of the Device memory.



2. Memory Bandwidth

Memory bandwidth limits the performance of a kernel when one or more memories in the GPU cannot provide data at the rate requested by the kernel. The results below indicate that the kernel is limited by the bandwidth available to the device memory.

2.1. Global Memory Alignment and Access Pattern

Memory bandwidth is used most efficiently when each global memory load and store has proper alignment and access pattern.

Optimization: Each entry below points to a global load or store within the kernel with an inefficient alignment or access pattern. For each load or store improve the alignment and access pattern of the memory access.

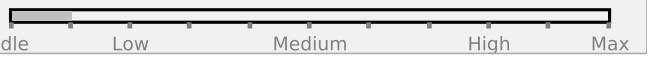
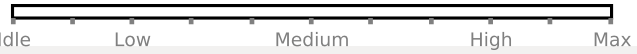
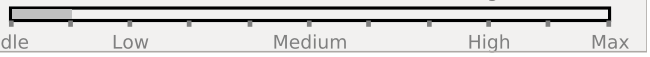
/home/giuseppe/Documents/myCUDA/ostacoli-2D/poligoni/improved/cuda_impr/const_memory/one_more/inside/inside.cu

Line 194	Global Load L2 Transactions/Access = 26.3, Ideal Transactions/Access = 4 [14417 L2 transactions for 548 total executions]
Line 194	Global Load L2 Transactions/Access = 26.3, Ideal Transactions/Access = 4 [14417 L2 transactions for 548 total executions]
Line 199	Global Store L2 Transactions/Access = 26.3, Ideal Transactions/Access = 4 [14417 L2 transactions for 548 total executions]
Line 199	Global Store L2 Transactions/Access = 26.3, Ideal Transactions/Access = 4 [14417 L2 transactions for 548 total executions]
Line 199	Global Store L2 Transactions/Access = 26.3, Ideal Transactions/Access = 4 [14417 L2 transactions for 548 total executions]
Line 199	Global Store L2 Transactions/Access = 26.3, Ideal Transactions/Access = 4 [14417 L2 transactions for 548 total executions]

2.2. GPU Utilization Is Limited By Memory Bandwidth

The following table shows the memory bandwidth used by this kernel for the various types of memory on the device. The table also shows the utilization of each memory type relative to the maximum throughput supported by the memory. The results show that the kernel's performance is potentially limited by the bandwidth available from one or more of the memories on the device.

Optimization: Try the following optimizations for the memory with high bandwidth utilization.
Shared Memory - If possible use 64-bit accesses to shared memory and 8-byte bank mode to achieved 2x throughput.
L2 Cache - Align and block kernel data to maximize L2 cache efficiency.
Unified Cache - Reallocate texture data to shared or global memory. Resolve alignment and access pattern issues for global loads and stores.
Device Memory - Resolve alignment and access pattern issues for global loads and stores.
System Memory (via PCIe) - Make sure performance critical data is placed in device or shared memory.

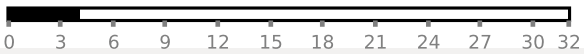
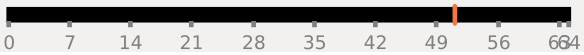
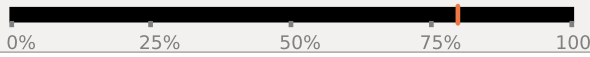
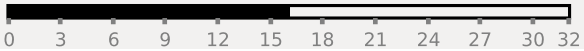
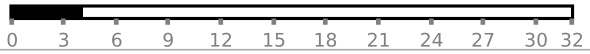
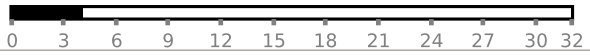
Transactions	Bandwidth	Utilization	
Shared Memory			
Shared Loads	0	0 B/s	
Shared Stores	0	0 B/s	
Shared Total	0	0 B/s	
L2 Cache			
Reads	23395	4.834 GB/s	
Writes	60464	12.494 GB/s	
Total	83859	17.328 GB/s	
Unified Cache			
Local Loads	0	0 B/s	
Local Stores	0	0 B/s	
Global Loads	34613	6.411 GB/s	
Global Stores	60458	12.493 GB/s	
Texture Reads	6576	1.359 GB/s	
Unified Total	101647	20.262 GB/s	
Device Memory			
Reads	16564	3.423 GB/s	
Writes	39049	8.069 GB/s	
Total	55613	11.491 GB/s	
System Memory			
[PCIe configuration: Gen2 x4, 5 Gbit/s]			
Reads	0	0 B/s	
Writes	5	1.033 MB/s	

3. Instruction and Memory Latency

Instruction and memory latency limit the performance of a kernel when the GPU does not have enough work to keep busy. The performance of latency-limited kernels can often be improved by increasing occupancy. Occupancy is a measure of how many warps the kernel has active on the GPU, relative to the maximum number of warps supported by the GPU. Theoretical occupancy provides an upper bound while achieved occupancy indicates the kernel's actual occupancy.

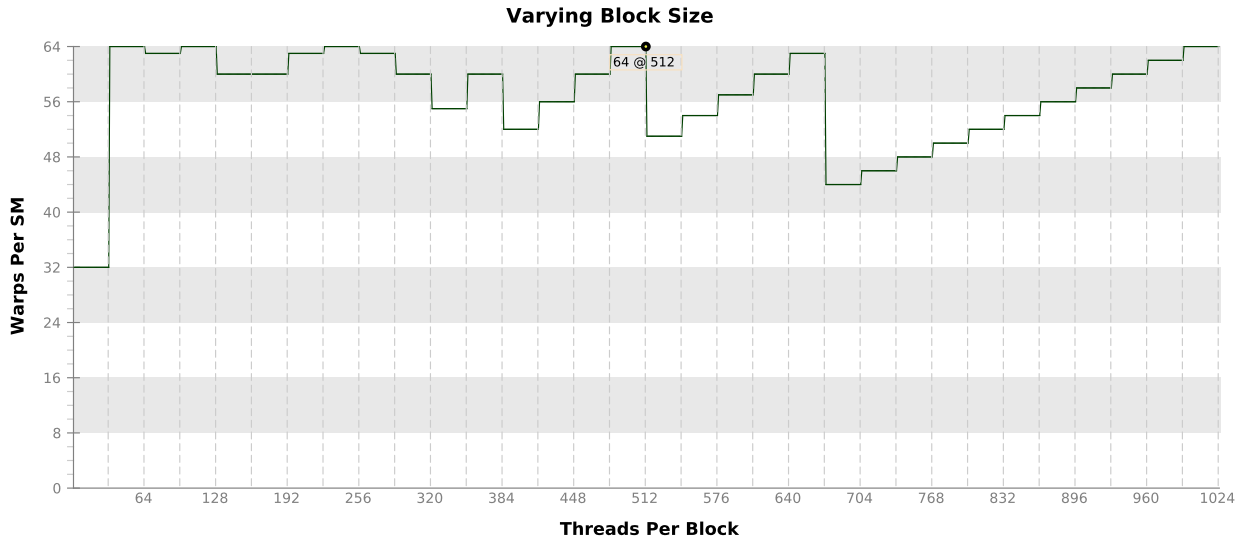
3.1. Occupancy Is Not Limiting Kernel Performance

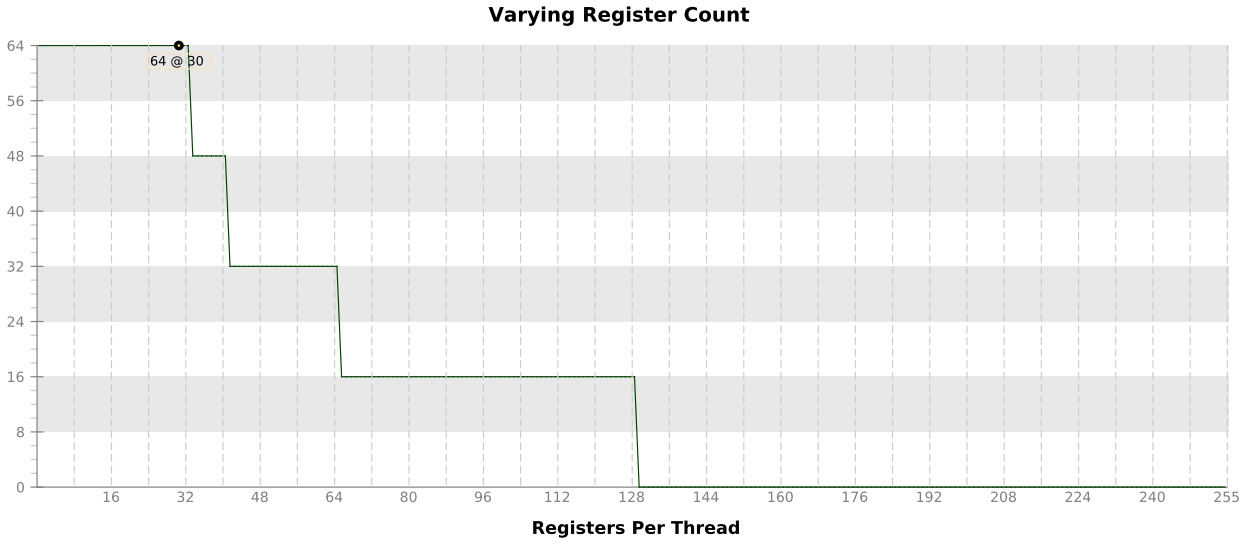
The kernel's block size, register usage, and shared memory usage allow it to fully utilize all warps on the GPU.

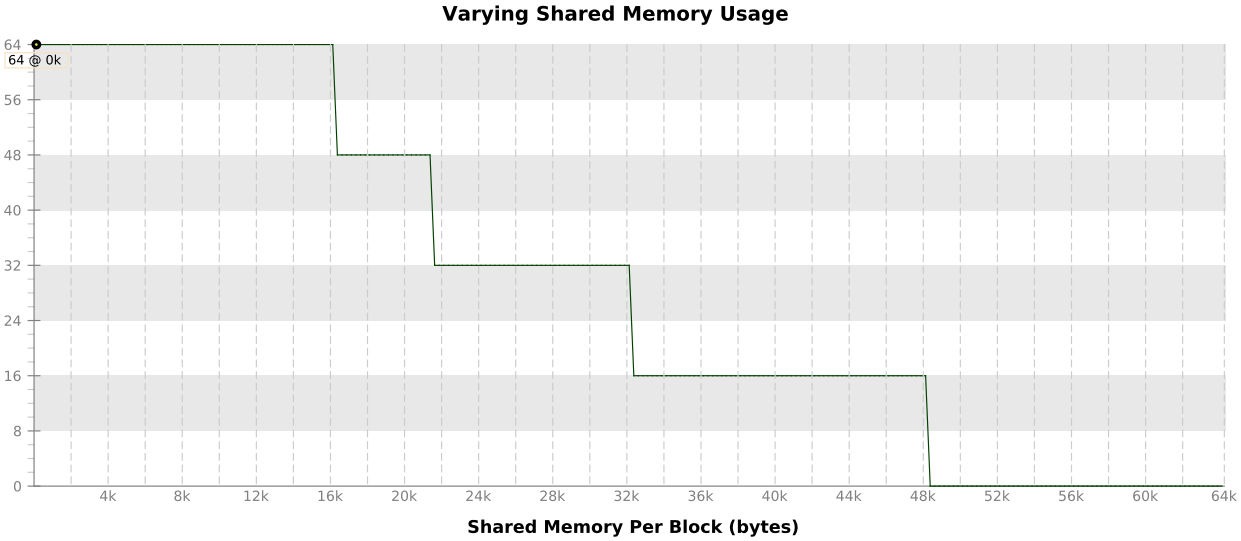
Variable	Achieved	Theoretical	Device Limit	Grid Size: [34,1,1] (34 blocks) Block Size: [512,1,1] (512 threads
Occupancy Per SM				
Active Blocks		4	32	
Active Warps	50.65	64	64	
Active Threads		2048	2048	
Occupancy	79.1%	100%	100%	
Warps				
Threads/Block		512	1024	
Warps/Block		16	32	
Block Limit		4	32	
Registers				
Registers/Thread		30	255	
Registers/Block		16384	65536	
Block Limit		4	32	
Shared Memory				
Shared Memory/Block		0	65536	
Block Limit			32	

3.2. Occupancy Charts

The following charts show how varying different components of the kernel will impact theoretical occupancy.


Varying Block Size
Warps Per SM
64
56
48
40
32
24
16
8
0
64 @ 512
64
128
192
256
320
384
448
512
576
640
704
768
832
896
960
1024
Threads Per Block


Varying Register Count
64
56
48
40
32
24
16
8
0
64 @ 30
16
32
48
64
80
96
112
128
144
160
176
192
208
224
240
255
Registers Per Thread


Varying Shared Memory Usage
64
56
48
40
32
24
16
8
0
64 @ 0k
4k
8k
12k
16k
20k
24k
28k
32k
36k
40k
44k
48k
52k
56k
60k
64k
Shared Memory Per Block (bytes)

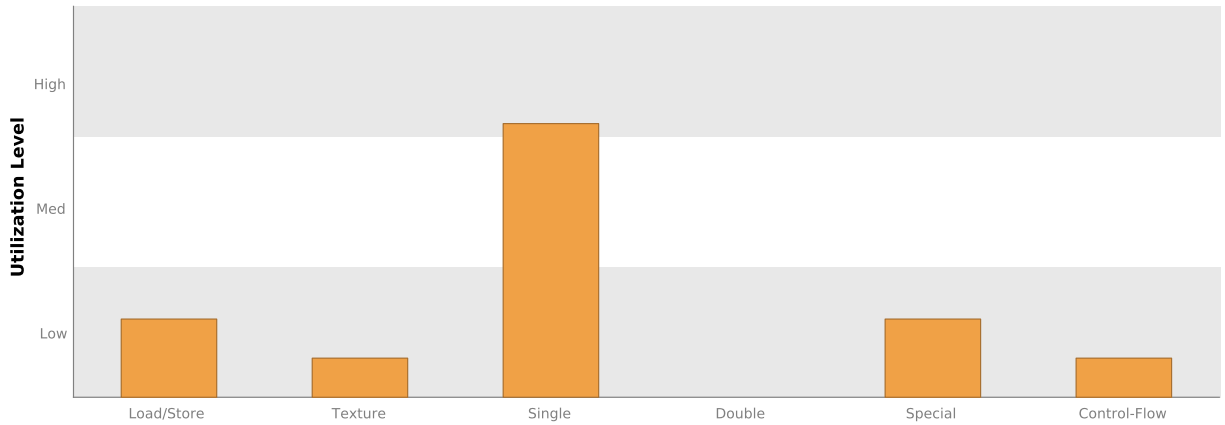
4. Compute Resources

GPU compute resources limit the performance of a kernel when those resources are insufficient or poorly utilized.

4.1. Function Unit Utilization

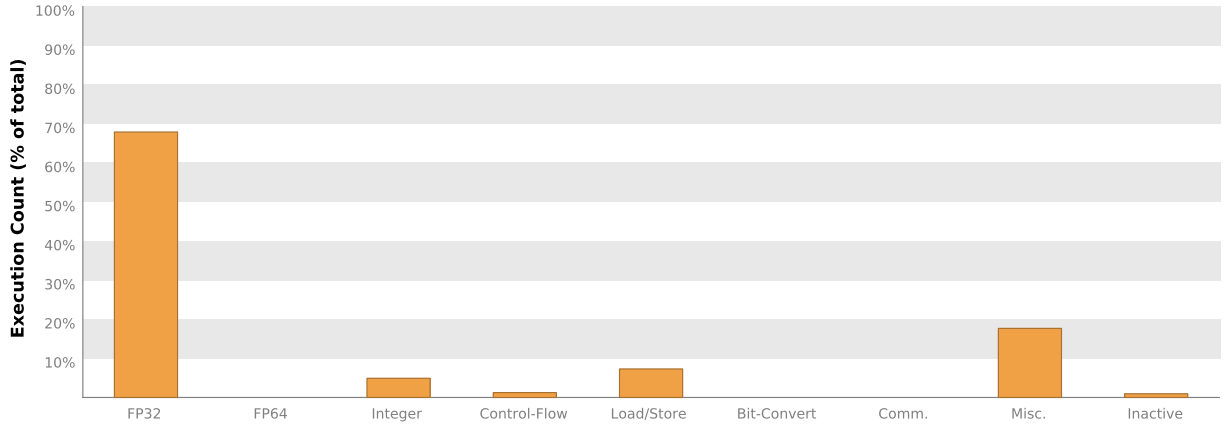
Different types of instructions are executed on different function units within each SM. Performance can be limited if a function unit is over-used by the instructions executed by the kernel. The following results show that the kernel's performance is not limited by overuse of any function unit.

Load/Store - Load and store instructions for shared and constant memory.
Texture - Load and store instructions for local, global, and texture memory.
Single - Single-precision integer and floating-point arithmetic instructions.
Double - Double-precision floating-point arithmetic instructions.
Special - Special arithmetic instructions such as sin, cos, popc, etc.
Control-Flow - Direct and indirect branches, jumps, and calls.



4.2. Instruction Execution Counts

The following chart shows the mix of instructions executed by the kernel. The instructions are grouped into classes and for each class the chart shows the percentage of thread execution cycles that were devoted to executing instructions in that class. The "Inactive" result shows the thread executions that did not execute any instruction because the thread was predicated or inactive due to divergence.



4.3. Floating-Point Operation Counts

The following chart shows the mix of floating-point operations executed by the kernel. The operations are grouped into classes and for each class the chart shows the percentage of thread execution cycles that were devoted to executing operations in that class. The results do not sum to 100% because non-floating-point operations executed by the kernel are not shown in this chart.