Analysis Report

fix_errors1(char*, Param*)

Duration	1.016 s (1,016,366,934 ns)
Grid Size	[256,1,1]
Block Size	[256,1,1]
Registers/Thread	85
Shared Memory/Block	0 B
Shared Memory Requested	48 KiB
Shared Memory Executed	48 KiB
Shared Memory Bank Size	4 B

[0] Tesla K20m

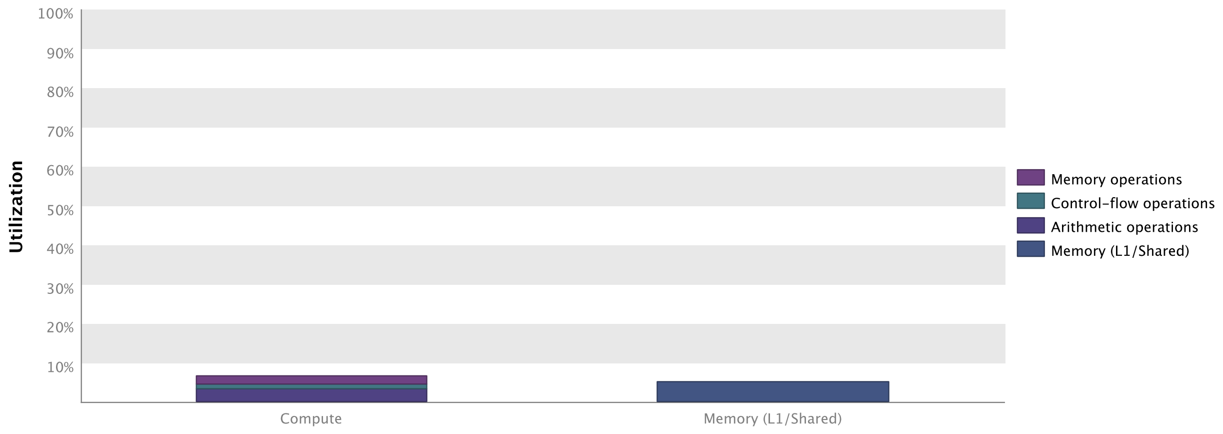
Compute Capability	3.5
Max. Threads per Block	1024
Max. Shared Memory per Block	48 KiB
Max. Registers per Block	65536
Max. Grid Dimensions	[2147483647, 65535, 65535]
Max. Block Dimensions	[1024, 1024, 64]
Max. Warps per Multiprocessor	64
Max. Blocks per Multiprocessor	16
Number of Multiprocessors	13
Multiprocessor Clock Rate	705.5 MHz
Concurrent Kernel	true
Max IPC	7
Threads per Warp	32
Global Memory Bandwidth	208 GB/s
Global Memory Size	4.687 GiB
Constant Memory Size	64 KiB
L2 Cache Size	1.25 MiB
Memcpy Engines	2
PCIe Generation	2
PCIe Link Rate	5 Gbit/s
PCIe Link Width	16

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 "fix_errors1" is most likely limited by instruction and memory latency. You should first examine the information in the "Instruction And Memory Latency" section to determine how it is limiting performance.

1.1. Kernel Performance Is Bound By Instruction And Memory Latency

This kernel exhibits low compute throughput and memory bandwidth utilization relative to the peak performance of "Tesla K20m". These utilization levels indicate that the performance of the kernel is most likely limited by the latency of arithmetic or memory operations. Achieved compute throughput and/or memory bandwidth below 60% of peak typically indicates latency issues.



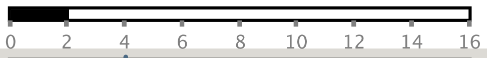
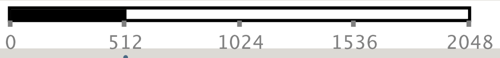
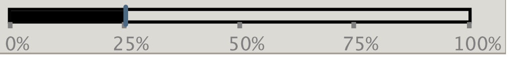
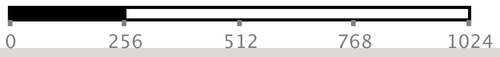
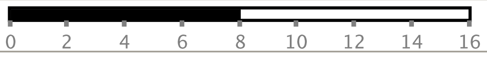
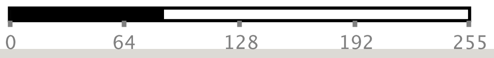
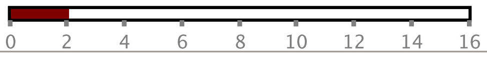
2. Instruction and Memory Latency

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. The results below indicate that occupancy can be improved by reducing the number of registers used by the kernel.

2.1. GPU Utilization Is Limited By Register Usage

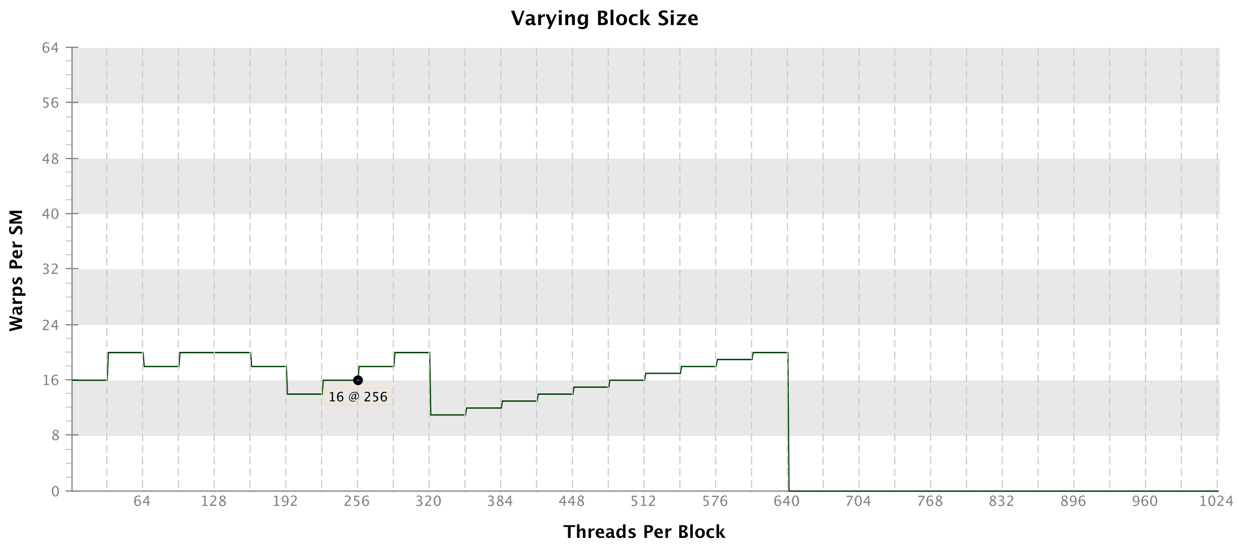
The kernel uses 85 registers for each thread (21760 registers for each block). This register usage is likely preventing the kernel from fully utilizing the GPU. Device "Tesla K20m" provides up to 65536 registers for each block. Because the kernel uses 21760 registers for each block each SM is limited to simultaneously executing 2 blocks (16 warps). Chart "Varying Register Count" below shows how changing register usage will change the number of blocks that can execute on each SM.

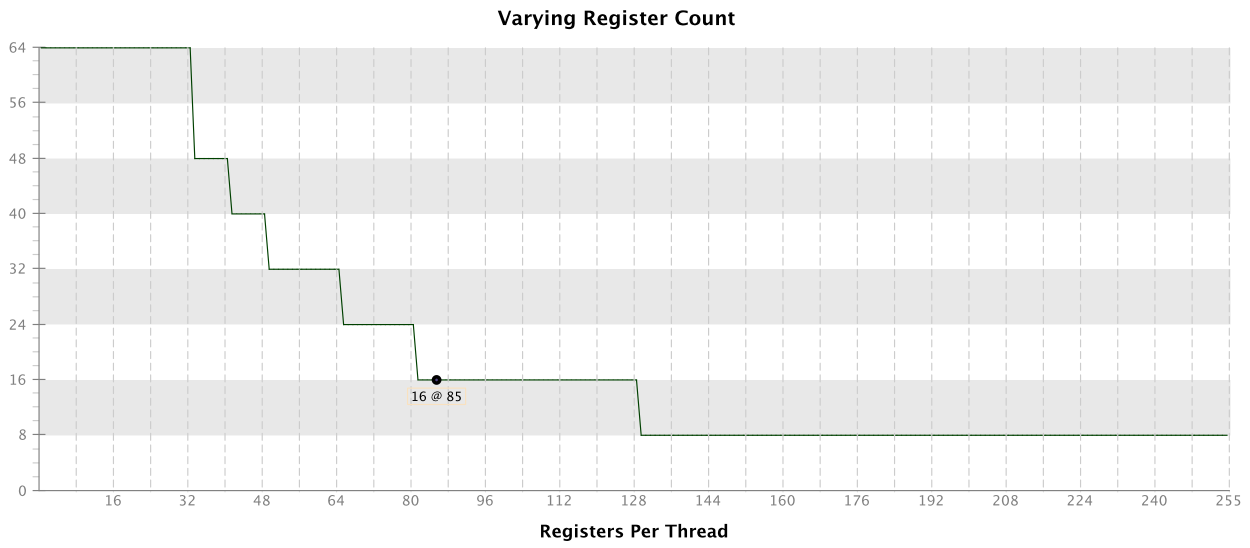
Optimization: Use the -maxrregcount flag or the __launch_bounds__ qualifier to decrease the number of registers used by each thread. This will increase the number of blocks that can execute on each SM.

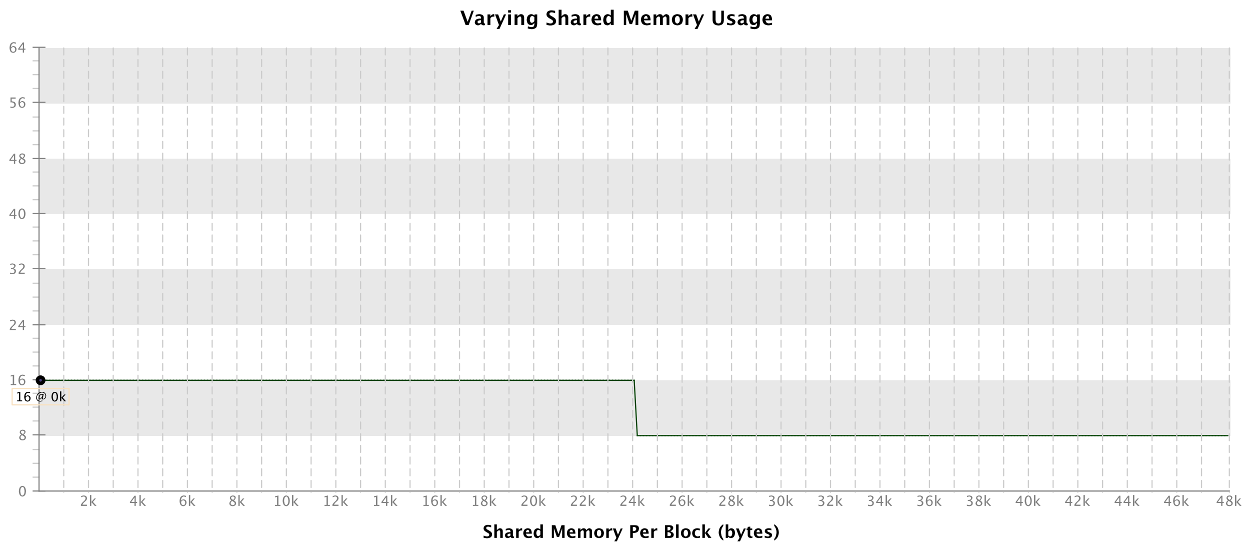
Variable	Achieved	Theoretical	Device Limit	Grid Size: [256,1,1] (256 blocks) Block Size: [256,1,1] (
Occupancy Per SM				
Active Blocks		2	16	
Active Warps	15.76	16	64	
Active Threads		512	2048	
Occupancy	24.6%	25%	100%	
Warps				
Threads/Block		256	1024	
Warps/Block		8	32	
Block Limit		8	16	
Registers				
Registers/Thread		85	255	
Registers/Block		22528	65536	
Block Limit		2	16	
Shared Memory				
Shared Memory/Block		0	49152	
Block Limit			16	

2.2. Occupancy Charts

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


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


Varying Register Count
Registers Per Thread
16 @ 85
0
8
16
24
32
40
48
56
64
16
32
48
64
80
96
112
128
144
160
176
192
208
224
240
255


Varying Shared Memory Usage
64
56
48
40
32
24
16
8
0
16 @ 0k
2k
4k
6k
8k
10k
12k
14k
16k
18k
20k
22k
24k
26k
28k
30k
32k
34k
36k
38k
40k
42k
44k
46k
48k
Shared Memory Per Block (bytes)

3. Compute Resources

GPU compute resources limit the performance of a kernel when those resources are insufficient or poorly utilized. Compute resources are used most efficiently when all threads in a warp have the same branching and predication behavior. The results below indicate that a significant fraction of the available compute performance is being wasted because branch and predication behavior is differing for threads within a warp.

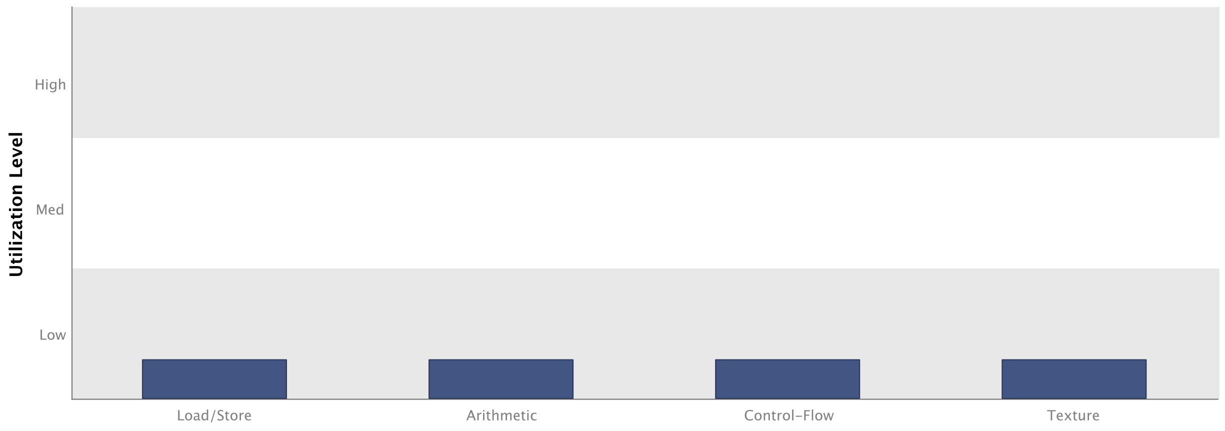
3.1. Low Warp Execution Efficiency

Warp execution efficiency is the average percentage of active threads in each executed warp. Increasing warp execution efficiency will increase utilization of the GPU's compute resources. The kernel's warp execution efficiency of 3.3% is less than 100% due to divergent branches and predicated instructions. If predicated instructions are not taken into account the warp execution efficiency for these kernels is 3.5%.

Optimization: Reduce the amount of intra-warp divergence and predication in the kernel.

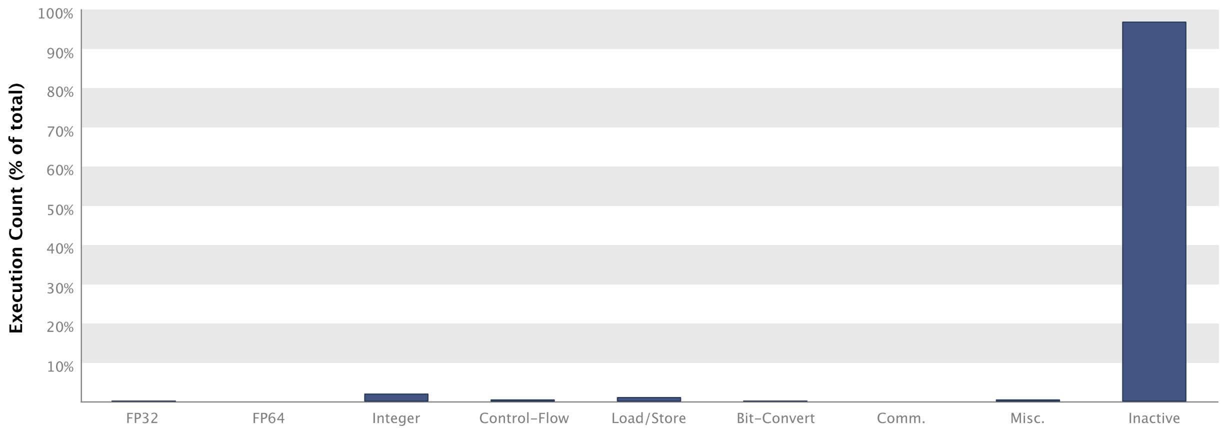
3.2. Function Unit Utilization

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 local, shared, global, constant, etc. memory.
Arithmetic - All arithmetic instructions including integer and floating-point add and multiply, logical and binary operations, etc.
Control-Flow - Direct and indirect branches, jumps, and calls.
Texture - Texture operations.



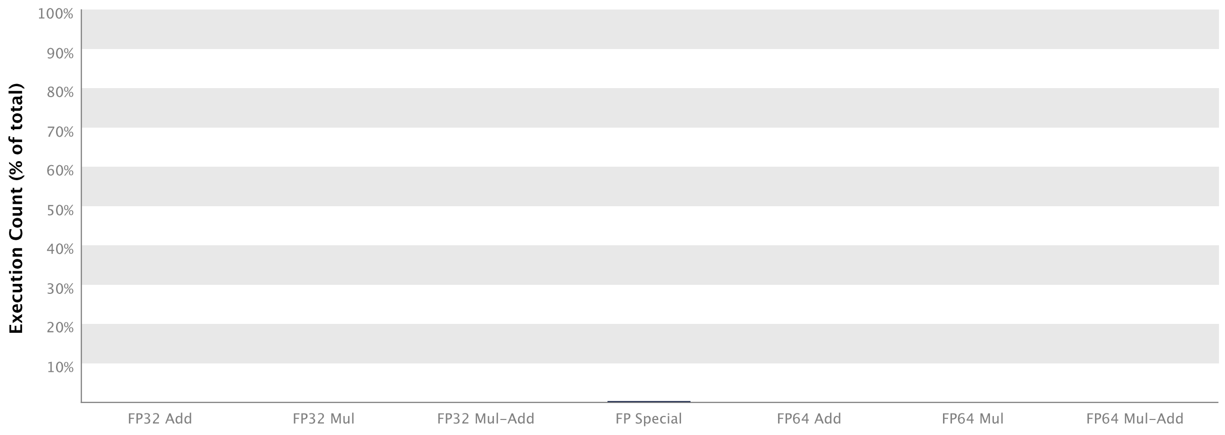
3.3. Instruction Execution Counts

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.



3.4. 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.

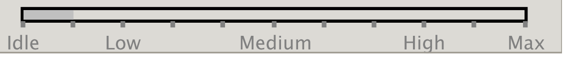
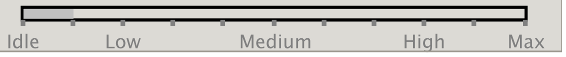
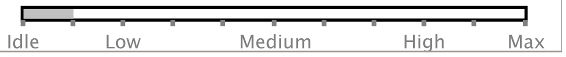
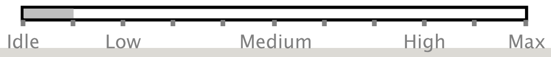
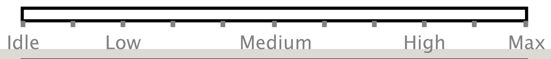
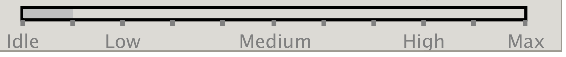


4. Memory Bandwidth

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.

4.1. Memory Bandwidth And Utilization

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.

	Transactions	Bandwidth	Utilization
L1/Shared Memory			
Local Loads	196863657	24.736 GB/s	
Local Stores	208211948	26.009 GB/s	
Shared Loads	0	0 B/s	
Shared Stores	0	0 B/s	
Global Loads	185550571	6.293 GB/s	
Global Stores	14225296	564.388 MB/s	
Atomic	0	0 B/s	
L1/Shared Total	604851472	57.603 GB/s	
L2 Cache			
L1 Reads	205407870	6.472 GB/s	
L1 Writes	24112603	759.795 MB/s	
Texture Reads	9001915	283.653 MB/s	
Atomic	0	0 B/s	
Noncoherent Reads	0	0 B/s	
Total	238522388	7.516 GB/s	
Texture Cache			
Reads	12113780	381.709 MB/s	
Device Memory			
Reads	24298031	765.638 MB/s	
Writes	39769228	1.253 GB/s	
Total	64067259	2.019 GB/s	
ECC Overhead	27969701	881.333 MB/s	
System Memory			
[PCIe configuration: Gen2 x16, 5 Gbit/s]			
Reads	0	0 B/s	
Writes	4	126 B/s	 Idle Low Medium High Max