Analysis Report

quadratic_difference_linear

Duration	38.708 ms (38,708,341 ns)
Grid Size	[7813,1,1]
Block Size	[576,1,1]
Registers/Thread	22
Shared Memory/Block	32.422 KiB
Shared Memory Requested	96 KiB
Shared Memory Executed	96 KiB
Shared Memory Bank Size	4 B

[0] GeForce GTX TITAN X

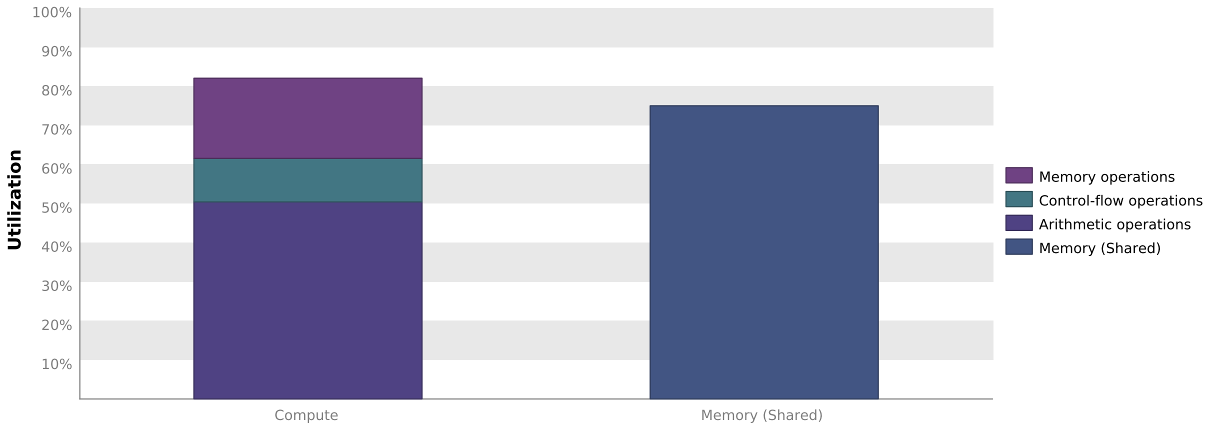
GPU UUID	GPU-274fb2c0-9ed5-60cc-1884-31c8645a8f30
Compute Capability	5.2
Max. Threads per Block	1024
Max. Shared Memory per Block	48 KiB
Max. Registers per Block	65536
Max. Grid Dimensions	[2147483647, 65535, 65535]
Max. Block Dimensions	[1024, 1024, 64]
Max. Warps per Multiprocessor	64
Max. Blocks per Multiprocessor	32
Single Precision FLOP/s	6.611 TeraFLOP/s
Double Precision FLOP/s	206.592 GigaFLOP/s
Number of Multiprocessors	24
Multiprocessor Clock Rate	1.076 GHz
Concurrent Kernel	true
Max IPC	6
Threads per Warp	32
Global Memory Bandwidth	336.48 GB/s
Global Memory Size	11.999 GiB
Constant Memory Size	64 KiB
L2 Cache Size	3 MiB
Memcpy Engines	2
PCIe Generation	3
PCIe Link Rate	8 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 "quadratic_difference_linear" 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 GTX TITAN X" 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 Shared 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 shared memory.

2.1. 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	844174868	2,785.663 GB/s	
Shared Stores	2030916	6.702 GB/s	
Shared Total	846205784	2,792.365 GB/s	Idle Low Medium High Max
L2 Cache			
Reads	16716979	13.791 GB/s	
Writes	13857124	11.432 GB/s	
Total	30574103	25.223 GB/s	Idle Low Medium High Max
Unified Cache			
Local Loads	374	308.537 kB/s	
Local Stores	374	308.537 kB/s	
Global Loads	16216092	6.702 GB/s	
Global Stores	13856744	11.431 GB/s	
Texture Reads	8124038	6.702 GB/s	
Unified Total	38197622	24.836 GB/s	Idle Low Medium High Max
Device Memory			
Reads	2295855	1.894 GB/s	
Writes	13890652	11.459 GB/s	
Total	16186507	13.353 GB/s	Idle Low Medium High Max
System Memory			
[PCIe configuration: Gen3 x16, 8 Gbit/s]			
Reads	0	0 B/s	Idle Low Medium High Max
Writes	5	4.124 kB/s	Idle Low Medium High Max

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. The results below indicate that occupancy can be improved by reducing the amount of shared memory used by the kernel.

3.1. GPU Utilization May Be Limited By Shared Memory Usage

Theoretical occupancy is less than 100% but is large enough that increasing occupancy may not improve performance. You can attempt the following optimization to increase the number of warps on each SM but it may not lead to increased performance.

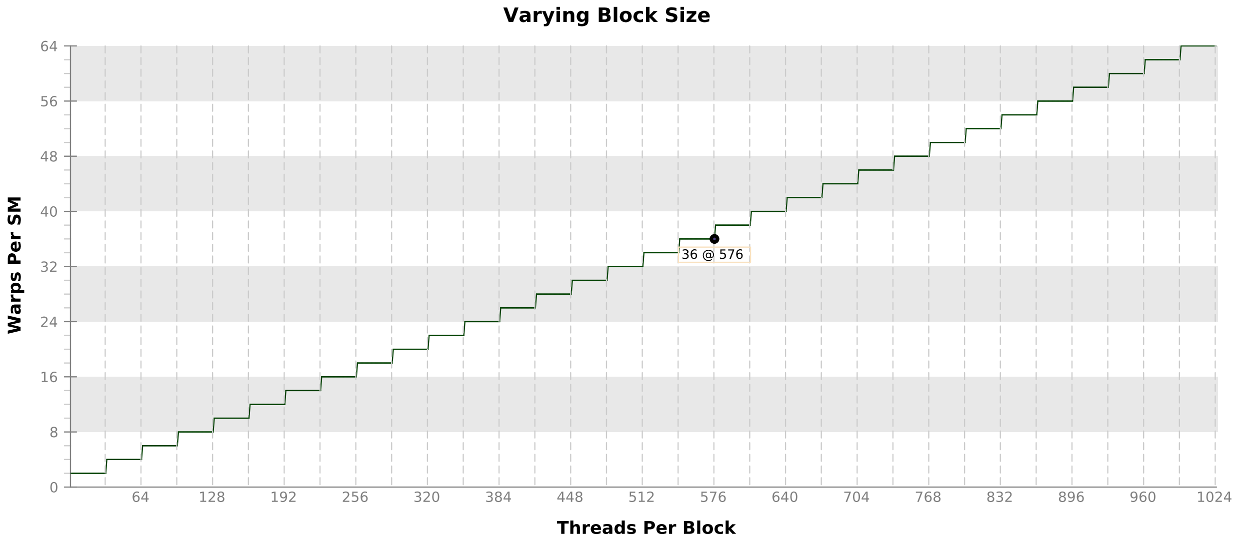
The kernel uses 32.422 KiB of shared memory for each block. This shared memory usage is likely preventing the kernel from fully utilizing the GPU. Device "GeForce GTX TITAN X" is configured to have 96 KiB of shared memory for each SM. Because the kernel uses 32.422 KiB of shared memory for each block each SM is limited to simultaneously executing 2 blocks (36 warps). Chart "Varying Shared Memory Usage" below shows how changing shared memory usage will change the number of blocks that can execute on each SM.

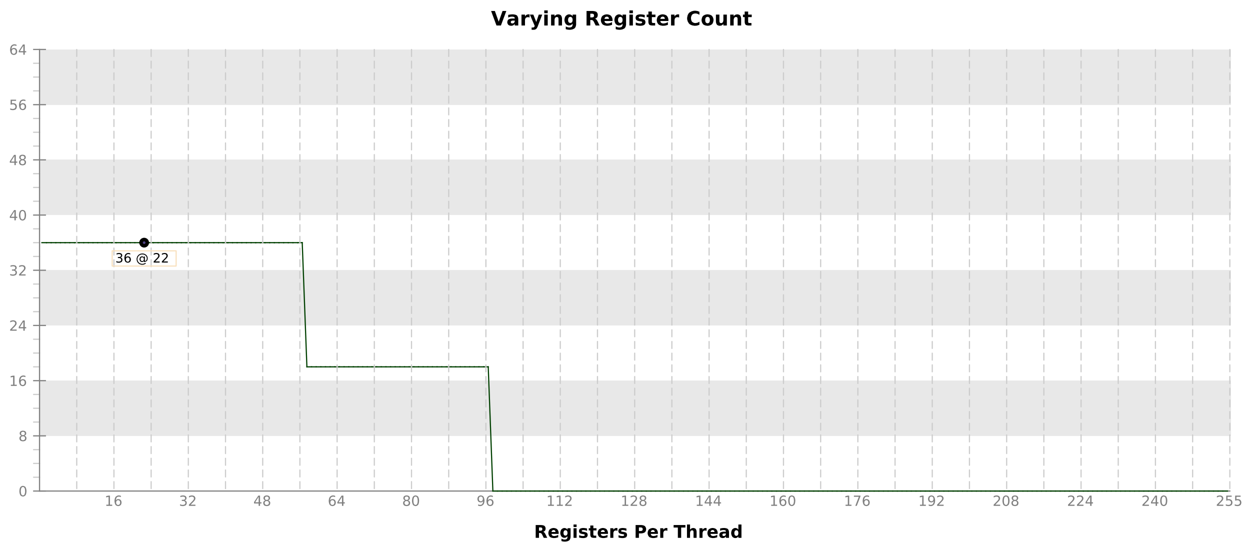
Optimization: Reduce shared memory usage to increase the number of blocks that can execute on each SM. You can also increase the number of blocks that can execute on each SM by increasing the amount of shared memory available to your kernel. You do this by setting the preferred cache configuration to "prefer shared".

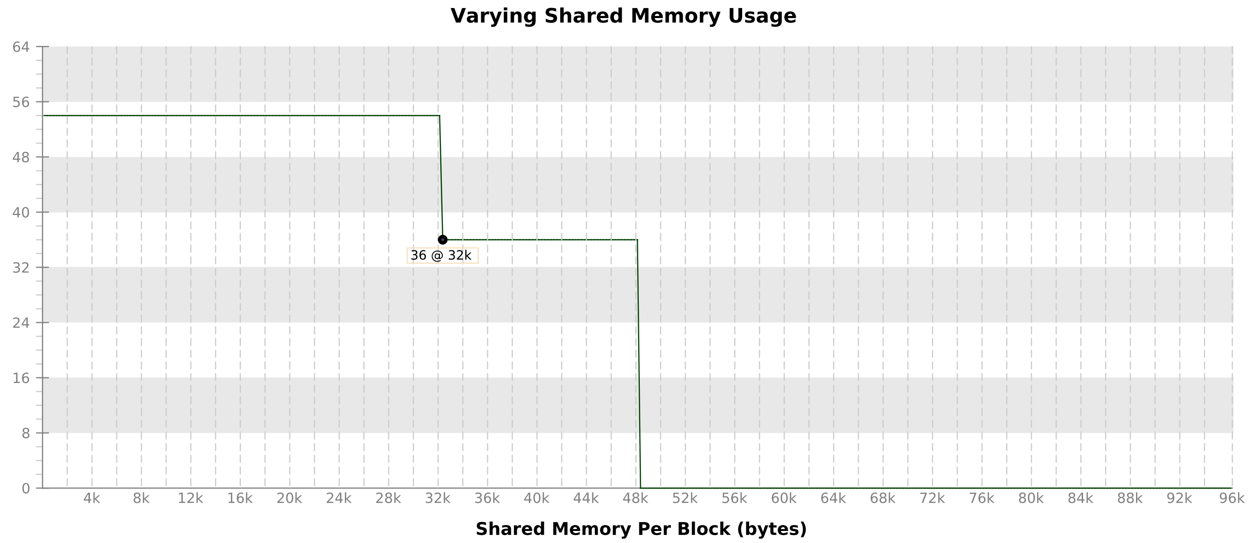
Variable	Achieved	Theoretical	Device Limit	Grid Size: [7813,1,1] (7813 blocks) Block Size: [576,1,
Occupancy Per SM				
Active Blocks		2	32	0 4 8 12 16 20 24 28 32
Active Warps	35.2	36	64	0 9 18 27 36 45 54 63 64
Active Threads		1152	2048	0 512 1024 1536 2048
Occupancy	55%	56.2%	100%	0% 25% 50% 75% 100%
Warps				
Threads/Block		576	1024	0 256 512 768 1024
Warps/Block		18	32	0 4 8 12 16 20 24 28 32
Block Limit		3	32	0 4 8 12 16 20 24 28 32
Registers				
Registers/Thread		22	255	0 64 128 192 255
Registers/Block		13824	65536	0 16k 32k 48k 64k
Block Limit		4	32	0 4 8 12 16 20 24 28 32
Shared Memory				
Shared Memory/Block		33200	98304	0 32k 64k 96k
Block Limit		2	32	0 4 8 12 16 20 24 28 32

3.2. Occupancy Charts

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






Varying Shared Memory Usage
64
56
48
40
32
24
16
8
0
36 @ 32k
4k
8k
12k
16k
20k
24k
28k
32k
36k
40k
44k
48k
52k
56k
60k
64k
68k
72k
76k
80k
84k
88k
92k
96k
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. 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.

4.1. Divergent Branches

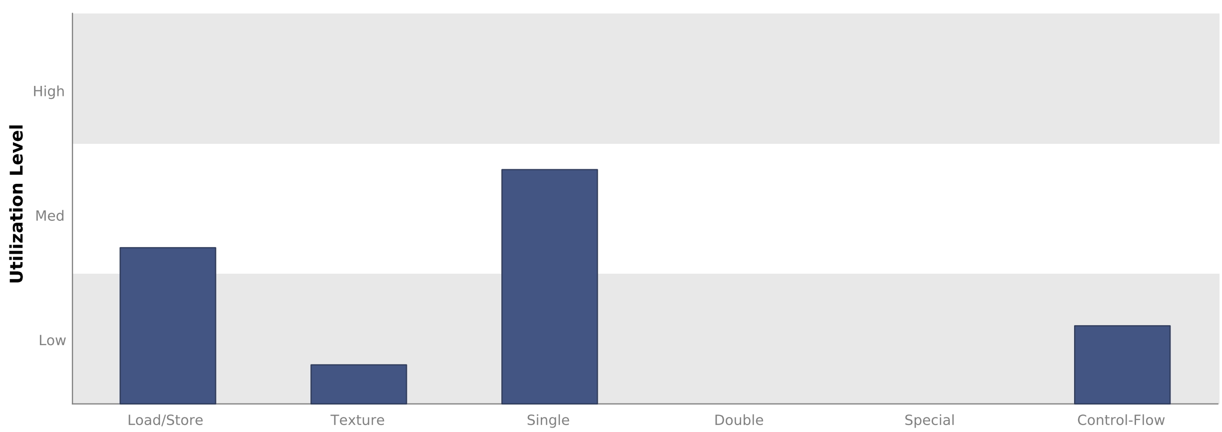
Compute resource are used most efficiently when all threads in a warp have the same branching behavior. When this does not occur the branch is said to be divergent. Divergent branches lower warp execution efficiency which leads to inefficient use of the GPU's compute resources.

Optimization: Each entry below points to a divergent branch within the kernel. For each branch reduce the amount of intra-warp divergence.

4.2. Function Unit Utilization

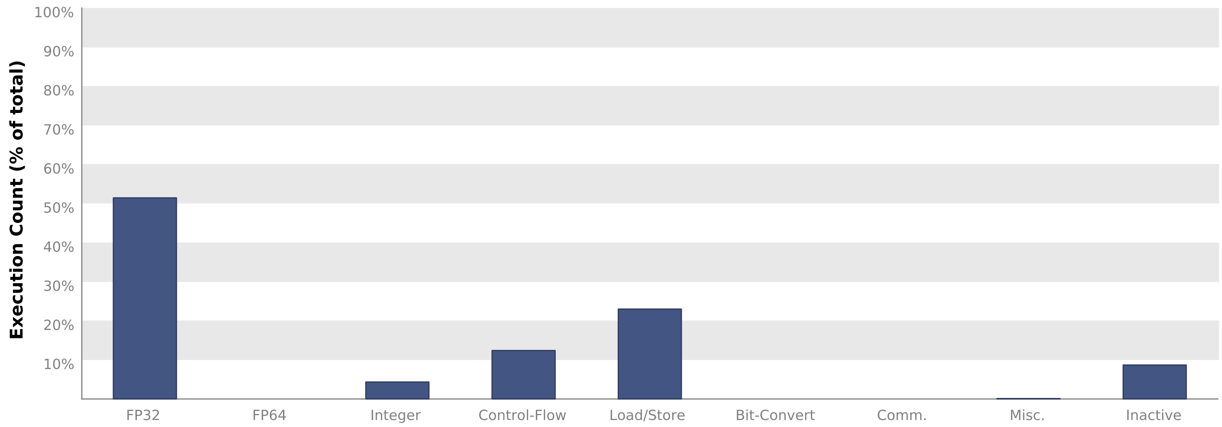
Different types of instructions are executed on different function units within each SM. Performance can be limited if a function unit is over-used by the instructions executed by the kernel. The following results show that the kernel's performance is not limited by overuse of any function unit.

Load/Store - Load and store instructions for 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.



4.3. 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.



4.4. 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.