Analysis Report

cudaPy__5F__5F_main_5F__5F__2E_quadratic_5F_difference_24_1_2E_array_28_

Duration	708.628 ms (708,628,221 ns)
Grid Size	[562500,94,1]
Block Size	[8,16,1]
Registers/Thread	28
Shared Memory/Block	496 B
Shared Memory Requested	96 KiB
Shared Memory Executed	96 KiB
Shared Memory Bank Size	4 B

[0] GeForce GTX TITAN X

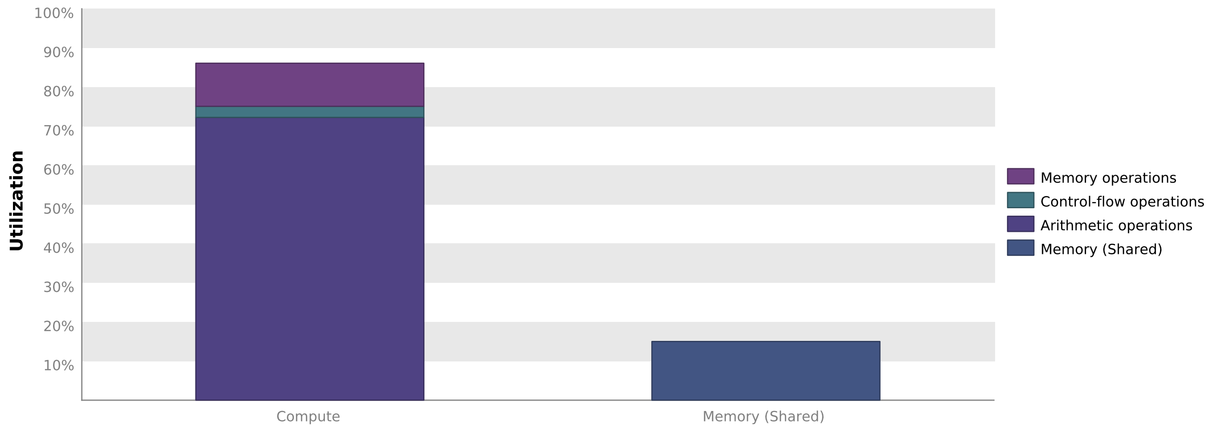
GPU UUID	GPU-0a4eae89-2e08-4851-bc8f-213711d16d38
Compute Capability	5.2
Max. Threads per Block	1024
Max. Shared Memory per Block	48 KiB
Max. Registers per Block	65536
Max. Grid Dimensions	[2147483647, 65535, 65535]
Max. Block Dimensions	[1024, 1024, 64]
Max. Warps per Multiprocessor	64
Max. Blocks per 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 "cudaPy__5F__5F_main_5F__5F_..." is most likely limited by compute. You should first examine the information in the "Compute Resources" section to determine how it is limiting performance.

1.1. Kernel Performance Is Bound By Compute

For device "GeForce GTX TITAN X" the kernel's memory utilization is significantly lower than its compute utilization. These utilization levels indicate that the performance of the kernel is most likely being limited by computation on the SMs.



2. 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.

2.1. 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 30.8% 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 31.5%.

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

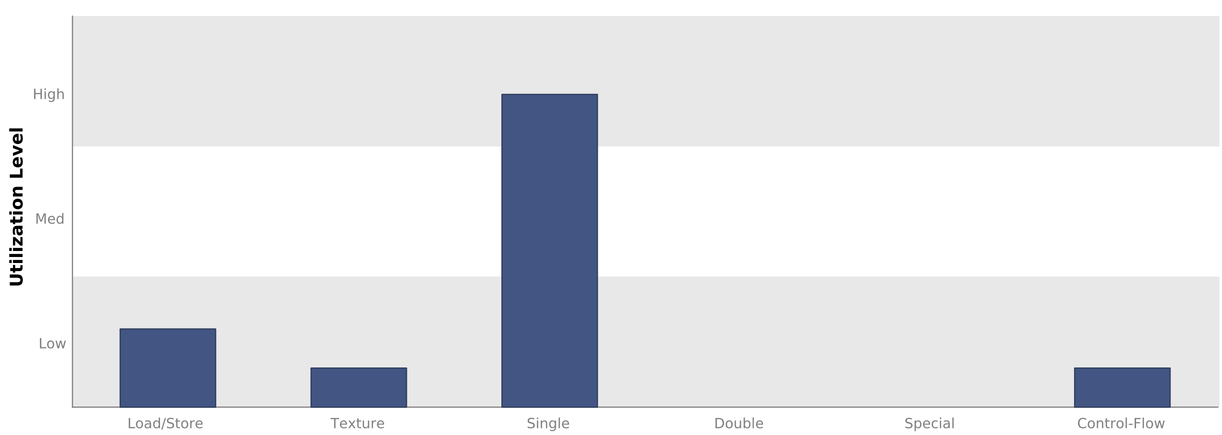
2.2. Divergent Branches

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.

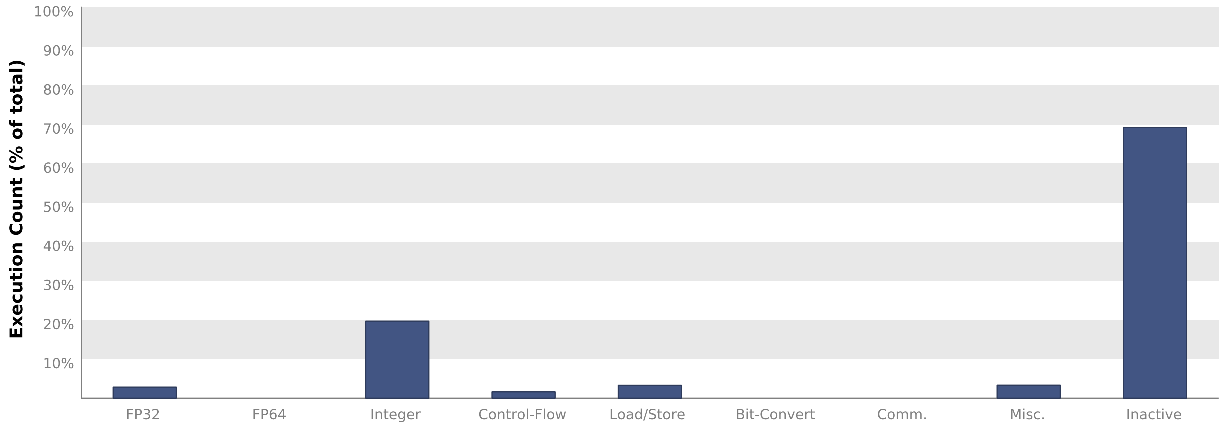
2.3. Function Unit Utilization

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.



2.4. 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.



2.5. 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.

3. 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.

3.1. 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	
Shared Memory			
Shared Loads	1687220248	310.578 GB/s	
Shared Stores	1903217248	350.338 GB/s	
Shared Total	3590437496	660.916 GB/s	Idle Low Medium High Max
L2 Cache			
Reads	2330180584	107.233 GB/s	
Writes	14262321	656.341 MB/s	
Total	2344442905	107.889 GB/s	Idle Low Medium High Max
Unified Cache			
Local Loads	0	0 B/s	
Local Stores	0	0 B/s	
Global Loads	4229363432	107.047 GB/s	
Global Stores	14262315	656.341 MB/s	
Texture Reads	3806434496	175.169 GB/s	
Unified Total	8050060243	282.873 GB/s	Idle Low Medium High Max
Device Memory			
Reads	213218239	9.812 GB/s	
Writes	14136448	650.548 MB/s	
Total	227354687	10.463 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	230 B/s	Idle Low Medium High Max

4. 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.

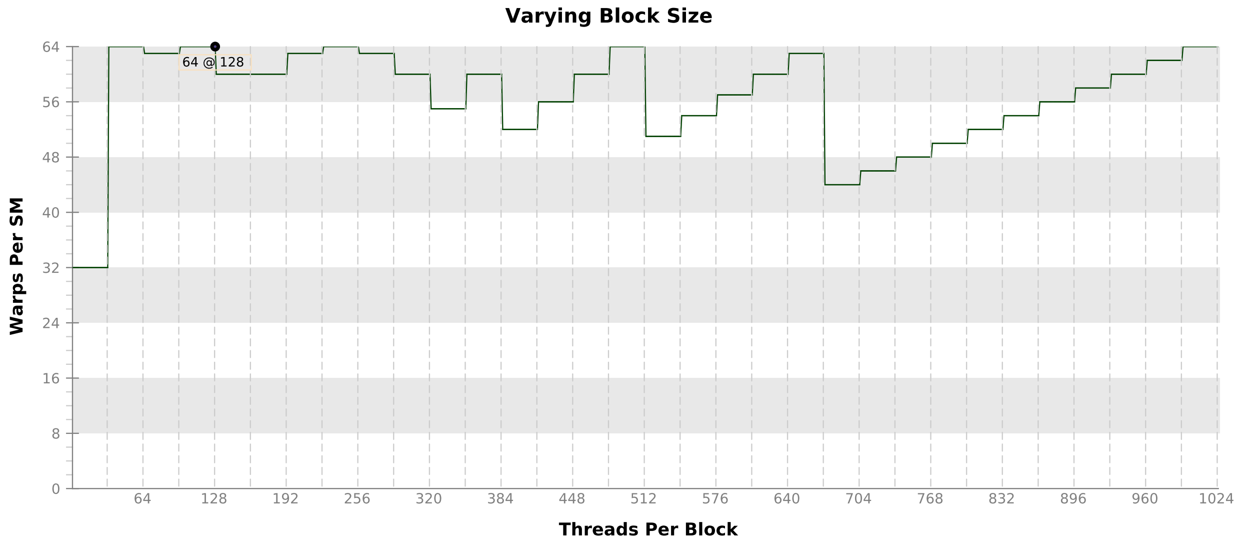
4.1. Occupancy Is Not Limiting Kernel Performance

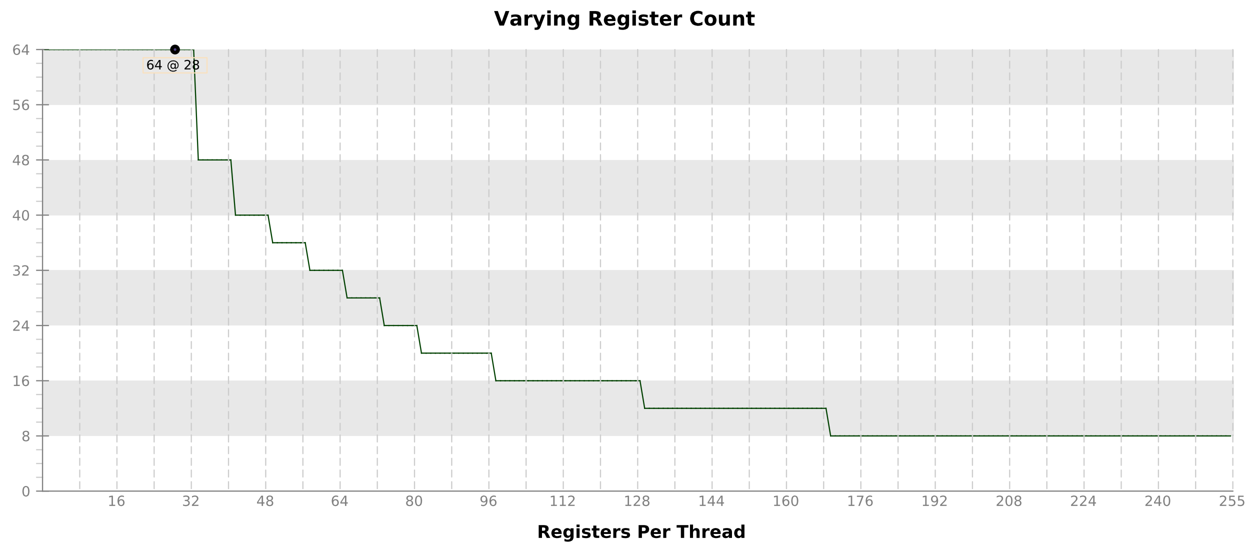
The kernel's block size, register usage, and shared memory usage allow it to fully utilize all warps on the GPU.

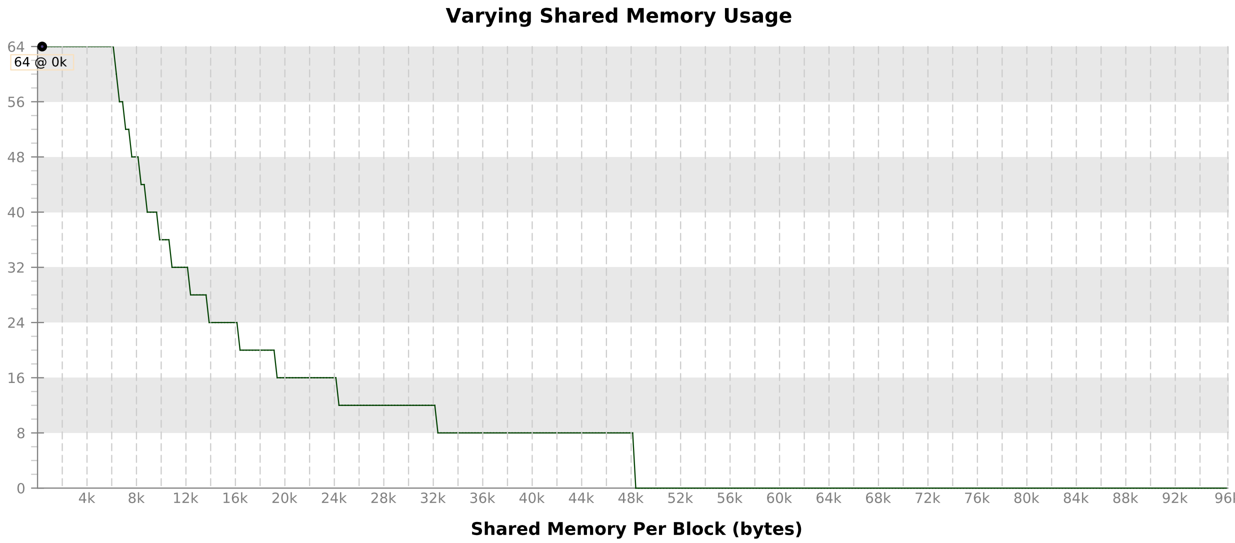
Variable	Achieved	Theoretical	Device Limit	Grid Size: [562500,94,1] (52875000 blocks) Block Size
Occupancy Per SM				
Active Blocks		16	32	0 4 8 12 16 20 24 28 32
Active Warps	61.91	64	64	0 9 18 27 36 45 54 63 64
Active Threads		2048	2048	0 512 1024 1536 2048
Occupancy	96.7%	100%	100%	0% 25% 50% 75% 100%
Warps				
Threads/Block		128	1024	0 256 512 768 1024
Warps/Block		4	32	0 4 8 12 16 20 24 28 32
Block Limit		16	32	0 4 8 12 16 20 24 28 32
Registers				
Registers/Thread		28	255	0 64 128 192 255
Registers/Block		4096	65536	0 16k 32k 48k 64k
Block Limit		16	32	0 4 8 12 16 20 24 28 32
Shared Memory				
Shared Memory/Block		496	98304	0 32k 64k 96k
Block Limit		192	32	0 4 8 12 16 20 24 28 32

4.2. Occupancy Charts

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


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


Varying Register Count
0
8
16
24
32
40
48
56
64
64 @ 28
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
68k
72k
76k
80k
84k
88k
92k
96k
Shared Memory Per Block (bytes)