Analysis Report

pollardKernel(unsigned int const *, unsigned int*, unsigned int*)

Duration	53,373 ms (53 373 092 ns)
Grid Size	[1024,1,1]
Block Size	[100,1,1]
Registers/Thread	46
Shared Memory/Block	0 B
Shared Memory Requested	16 KiB
Shared Memory Executed	16 KiB
Shared Memory Bank Size	4 B

[0] GeForce GTX 780 Ti

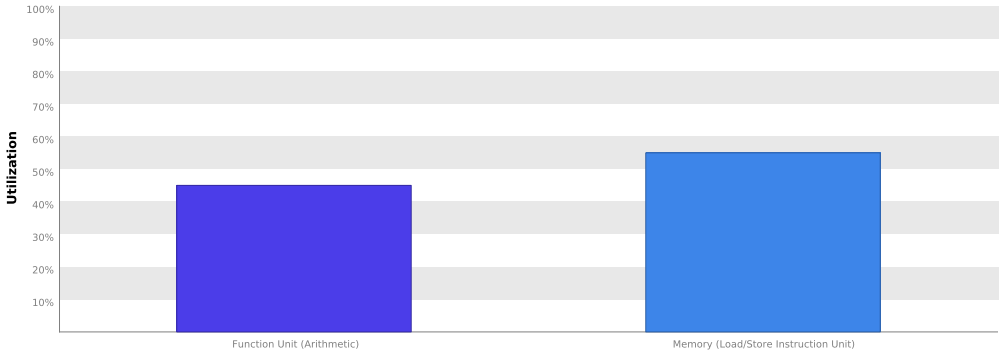
GPU UUID	GPU-6dfac189-49e4-701d-5da2-17d8a2ae1f8b
Compute Capability	3.5
Max. Threads per Block	1024
Max. Threads per Multiprocessor	2048
Max. Shared Memory per Block	48 KiB
Max. Shared Memory per Multiprocessor	48 KiB
Max. Registers per Block	65536
Max. Registers per Multiprocessor	65536
Max. Grid Dimensions	[2147483647, 65535, 65535]
Max. Block Dimensions	[1024, 1024, 64]
Max. Warps per Multiprocessor	64
Max. Blocks per Multiprocessor	16
Single Precision FLOP/s	6,172 TeraFLOP/s
Double Precision FLOP/s	257,16 GigaFLOP/s
Number of Multiprocessors	15
Multiprocessor Clock Rate	1,071 GHz
Concurrent Kernel	true
Max IPC	7
Threads per Warp	32
Global Memory Bandwidth	336 GB/s
Global Memory Size	2,95 GiB
Constant Memory Size	64 KiB
L2 Cache Size	1,5 MiB
Memcpy Engines	1
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 "pollardKernel" 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 "GeForce GTX 780 Ti". 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 May Be Limited By Register 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.

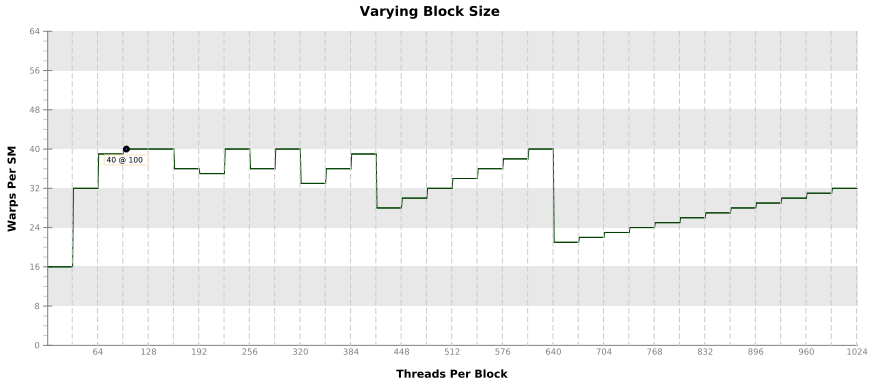
The kernel uses 46 registers for each thread (4600 registers for each block). This register usage is likely preventing the kernel from fully utilizing the GPU. Device "GeForce GTX 780 Ti" provides up to 65536 registers for each block. Because the kernel uses 4600 registers for each block each SM is limited to simultaneously executing 10 blocks (40 warps). Chart "Varying Register Count" below shows how changing register usage will change the number of blocks that can execute on each SM.

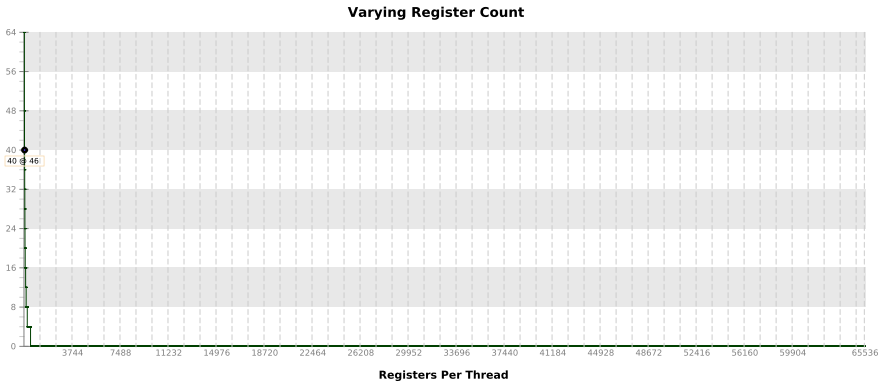
Optimization: Use the -maxrregcount flag or the __launch_bounds__ qualifier to decrease the number of registers used by each thread. This will increase the number of blocks that can execute on each SM. On devices with Compute Capability 5.2 turning global cache off can increase the occupancy limited by register usage.

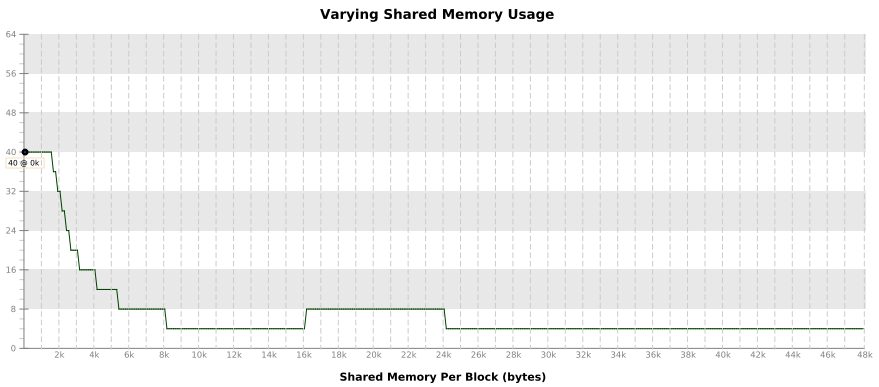
Variable	Achieved	Theoretical	Device Limit	Grid Size: [1024,1,1] (1024 blocks) Block Size: [100,1,1] (100 thr
Occupancy Per SM				
Active Blocks		10	16	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Active Warps	35,64	40	64	0 7 14 21 28 35 42 49 56 64
Active Threads		1280	2048	0 256 512 768 1024 1280 1536 1792 2048
Occupancy	55,7%	62,5%	100%	0% 25% 50% 75% 100%
Warps				
Threads/Block		100	1024	0 128 256 384 512 640 768 896 1024
Warps/Block		4	32	0 3 6 9 12 15 18 21 24 27 30 32
Block Limit		16	16	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Registers				
Registers/Thread		46	65536	0 8192 16384 24576 32768 40960 49152 57344 65536
Registers/Block		6144	65536	0 16k 32k 48k 64k
Block Limit		10	16	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Shared Memory				
Shared Memory/Block		0	16384	0 4k 8k 12k 16k
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
64
56
48
40
32
24
16
8
0
40 @ 100
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
40 @ 46
3744
7488
11232
14976
18720
22464
26208
29952
33696
37440
41184
44928
48672
52416
56160
59904
65536
Registers Per Thread


Varying Shared Memory Usage
64
56
48
40
32
24
16
8
0
40 @ 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. Kernel Profile - Instruction Execution

The Kernel Profile - Instruction Execution shows the execution count, inactive threads, and predicated threads for each source and assembly line of the kernel. Using this information you can pinpoint portions of your kernel that are making inefficient use of compute resource due to divergence and predication.

Examine portions of the kernel that have high execution counts and inactive or predicated threads to identify optimization opportunities.

Cuda Fuctions :

pollardKernel(unsigned int const *, unsigned int*, unsigned int*)

Maximum instruction execution count in assembly: 12478876
Average instruction execution count in assembly: 716726
Instructions executed for the kernel: 1811166793
Thread instructions executed for the kernel: 31660047474
Non-predicated thread instructions executed for the kernel: 29193971935
Warp non-predicated execution efficiency of the kernel: 50,4%
Warp execution efficiency of the kernel: 54,6%

3.2. 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 maximum warp execution efficiency is 78,1% because the number of threads per block is not a multiple of the warp size.

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

3.3. Divergent Branches

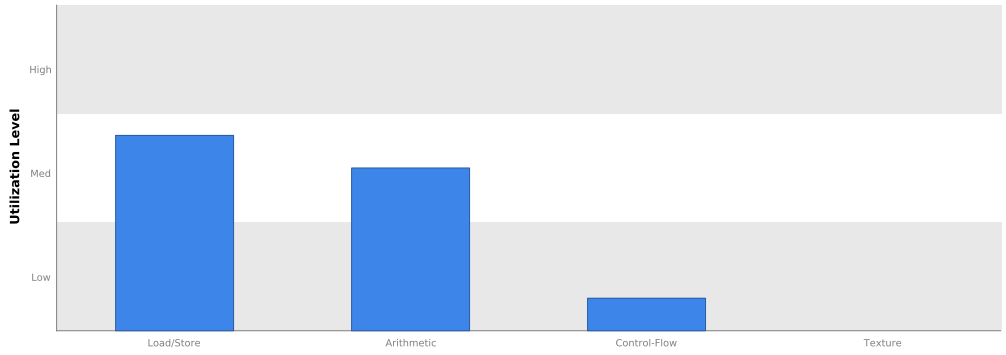
Compute resource are used most efficiently when all threads in a warp have the same branching behavior. When this does not occur the branch is said to be divergent. Divergent branches lower warp execution efficiency which leads to inefficient use of the GPU's compute resources.

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

3.4. Function Unit Utilization

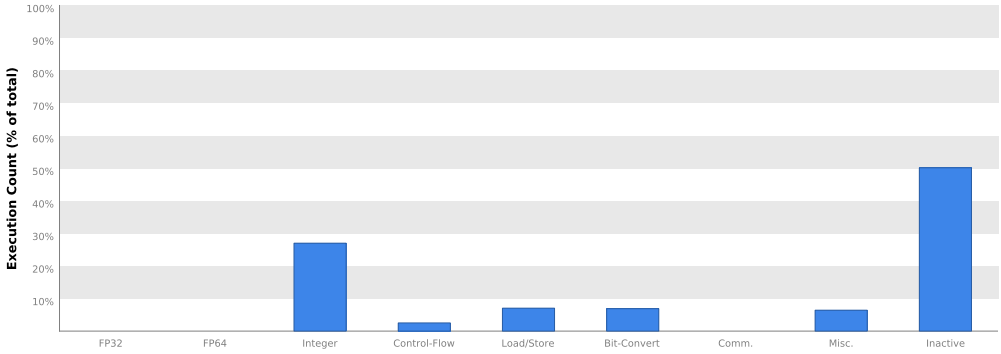
Different types of instructions are executed on different function units within each SM. Performance can be limited if a function unit is over-used by the instructions executed by the kernel. The following results show that the kernel's performance is not limited by overuse of any function unit.

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.



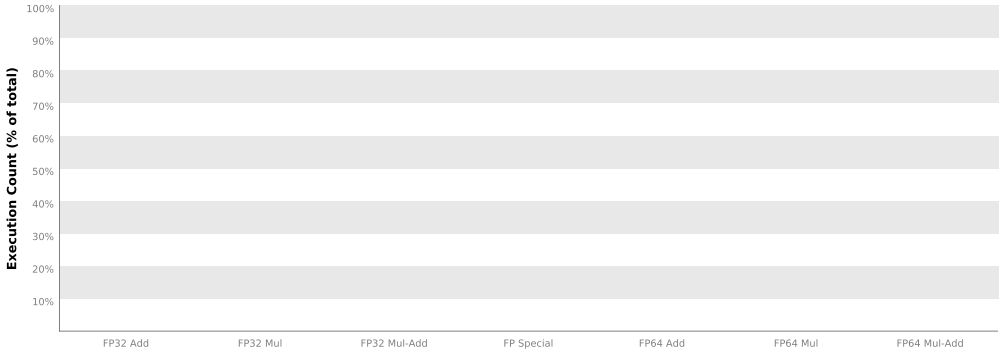
3.5. Instruction Execution Counts

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.



3.6. 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.

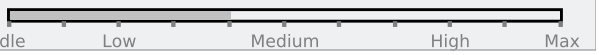
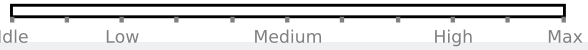


4. Memory Bandwidth

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	120565607	290,652 GB/s	
Local Stores	86410989	205,463 GB/s	
Shared Loads	0	0 B/s	
Shared Stores	0	0 B/s	
Global Loads	235913665	246,464 GB/s	
Global Stores	25140134	28,874 GB/s	
Atomic	0	0 B/s	
L1/Shared Total	468030395	771,454 GB/s	
L2 Cache			
L1 Reads	420221310	253,267 GB/s	
L1 Writes	63570090	38,314 GB/s	
Texture Reads	0	0 B/s	
Noncoherent Reads	0	0 B/s	
Atomic	0	0 B/s	
Total	483791400	291,581 GB/s	
Texture Cache			
Reads	0	0 B/s	
Device Memory			
Reads	238372	143,667 MB/s	
Writes	60771658	36,627 GB/s	
Total	61010030	36,771 GB/s	
System Memory			
[PCIe configuration: Gen3 x16, 8 Gbit/s]			
Reads	0	0 B/s	
Writes	4	2,41 kB/s