

Analysis Report

md5Kernel (MD5Task)

Duration	2.783 ms (2,783,332 ns)
Grid Size	[16,1,1]
Block Size	[128,1,1]
Registers/Thread	32
Shared Memory/Block	532 B
Shared Memory Requested	48 KiB
Shared Memory Bank Size	4 B

[0] GK20A

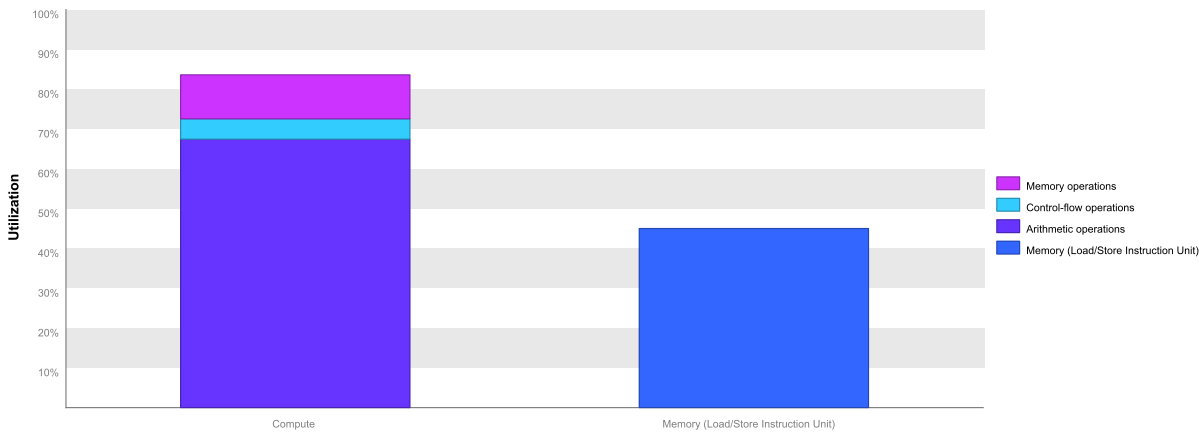
Compute Capability	3.2
Max. Threads per Block	1024
Max. Shared Memory per Block	48 KiB
Max. Registers per Block	32768
Max. Grid Dimensions	[2147483647, 65535, 65535]
Max. Block Dimensions	[1024, 1024, 64]
Max. Warps per Multiprocessor	64
Max. Blocks per Multiprocessor	16
Single Precision FLOP/s	327.168 GigaFLOP/s
Double Precision FLOP/s	13.632 GigaFLOP/s
Number of Multiprocessors	1
Multiprocessor Clock Rate	852 MHz
Concurrent Kernel	true
Max IPC	7
Threads per Warp	32
Global Memory Bandwidth	14.784 GB/s
Global Memory Size	1.848 GiB
Constant Memory Size	64 KiB
L2 Cache Size	128 KiB
Memcpy Engines	1

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 "md5Kernel" 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 "GK20A" 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. 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 Functions :

md5Kernel(MD5Task)

Maximum instruction execution count in assembly: 115200

Average instruction execution count in assembly: 14741

Instructions executed for the kernel: 10319376

Thread instructions executed for the kernel: 330198688

Non-predicated thread instructions executed for the kernel: 320772720

Warp non-predicated execution efficiency of the kernel: 97.1%

Warp execution efficiency of the kernel: 100.0%

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.

</home/mark/project/advent-of-cuda/day4/day4.cu>

Line 48	Divergence = 25% [16 divergent executions out of 64 total executions]
Line 48	Divergence = 25% [16 divergent executions out of 64 total executions]

2.3. Function Unit Utilization

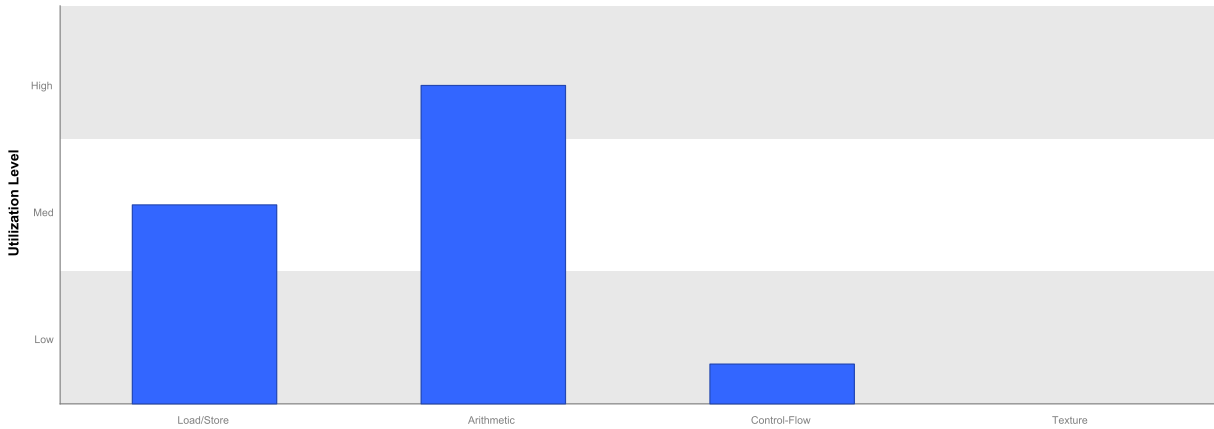
Different types of instructions are executed on different function units within each SM. Performance can be limited if a function unit is over-used by the instructions executed by the kernel. The following results show that the kernel's performance is not limited by overuse of any function unit.

Load/Store - Load and store instructions for local, shared, global, constant, etc. memory.

Arithmetic - All arithmetic instructions including integer and floating-point add and multiply, logical and binary operations, etc.

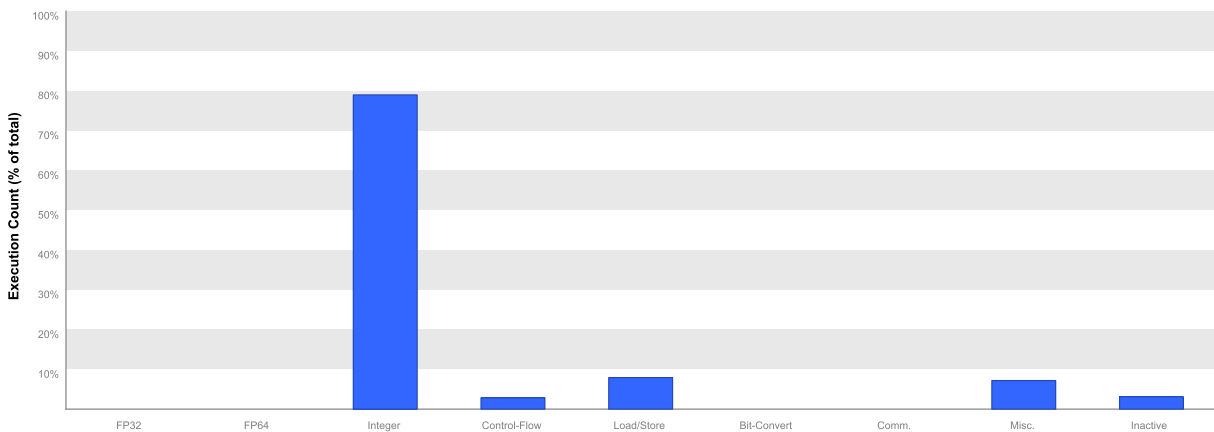
Control-Flow - Direct and indirect branches, jumps, and calls.

Texture - Texture operations.



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.



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.


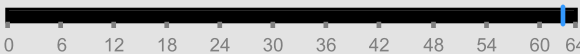

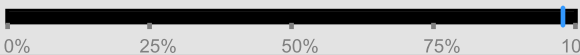

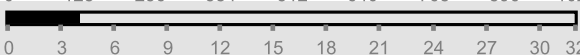






Transactions	Bandwidth	Utilization	
L2 Cache			
Reads	11143	81.1 MB/s	
Writes	21	152.839 kB/s	
Total	11164	81.252 MB/s	
Texture Cache			
Reads	0	0 B/s	

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: [16,1,1] (16 blocks) Block Size: [128,1,1] (128 threads)
Occupancy Per SM				
Active Blocks		16	16	
Active Warps	62.27	64	64	
Active Threads		2048	2048	
Occupancy	97.3%	100%	100%	
Warps				
Threads/Block		128	1024	
Warps/Block		4	32	
Block Limit		16	16	
Registers				
Registers/Thread		32	255	
Registers/Block		4096	65536	
Block Limit		16	16	
Shared Memory				
Shared Memory/Block		532		
Block Limit		64	16	

4.2. Occupancy Charts

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

