

Analysis Report

matmult_gpu2Kernel(int, int, int, double*, double*, double*)

Duration	2.81313 s (2,813,128,136 ns)
Grid Size	[313,313,1]
Block Size	[16,16,1]
Registers/Thread	32
Shared Memory/Block	0 B
Shared Memory Requested	96 KiB
Shared Memory Executed	96 KiB
Shared Memory Bank Size	4 B

[0] Tesla V100-SXM2-32GB

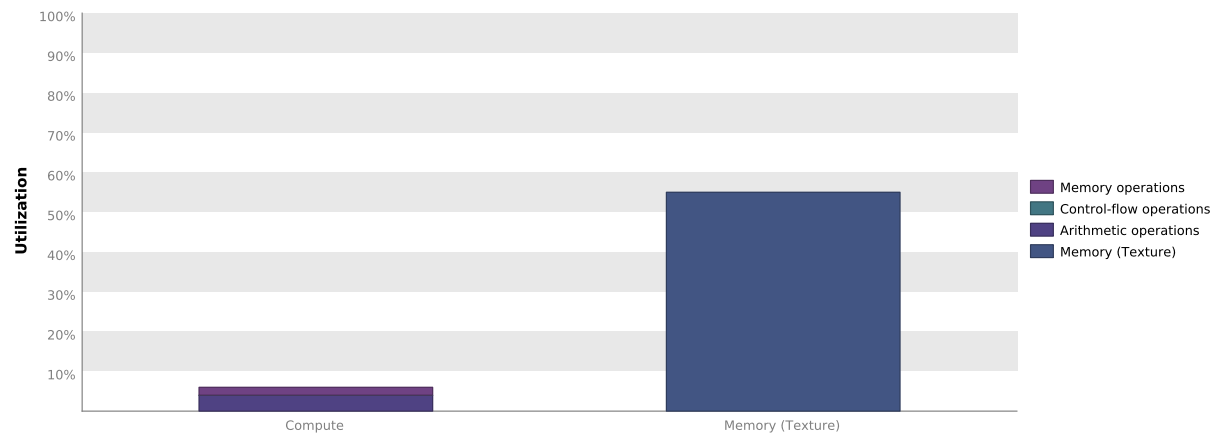
GPU UUID	GPU-c2fbdc1c-622a-33f0-430e-76df65ca5880
Compute Capability	7.0
Max. Threads per Block	1024
Max. Threads per Multiprocessor	2048
Max. Shared Memory per Block	48 KiB
Max. Shared Memory per Multiprocessor	96 KiB
Max. Registers per Block	65536
Max. Registers per Multiprocessor	65536
Max. Grid Dimensions	[2147483647, 65535, 65535]
Max. Block Dimensions	[1024, 1024, 64]
Max. Warps per Multiprocessor	64
Max. Blocks per Multiprocessor	32
Half Precision FLOP/s	31.334 TeraFLOP/s
Single Precision FLOP/s	15.667 TeraFLOP/s
Double Precision FLOP/s	7.834 TeraFLOP/s
Number of Multiprocessors	80
Multiprocessor Clock Rate	1.53 GHz
Concurrent Kernel	true
Max IPC	4
Threads per Warp	32
Global Memory Bandwidth	898.048 GB/s
Global Memory Size	31.719 GiB
Constant Memory Size	64 KiB
L2 Cache Size	6 MiB
Memcpy Engines	5
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 "matmult_gpu2Kernel" 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 V100-SXM2-32GB". 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.

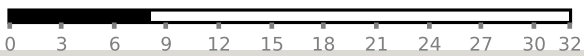


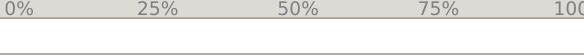
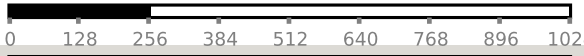
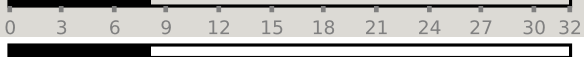

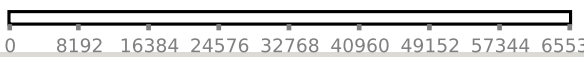

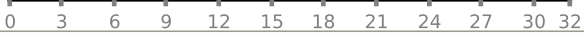
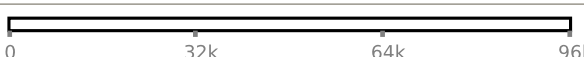
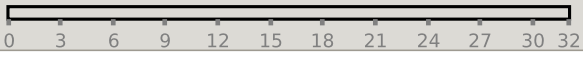


2. Instruction and Memory Latency

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.

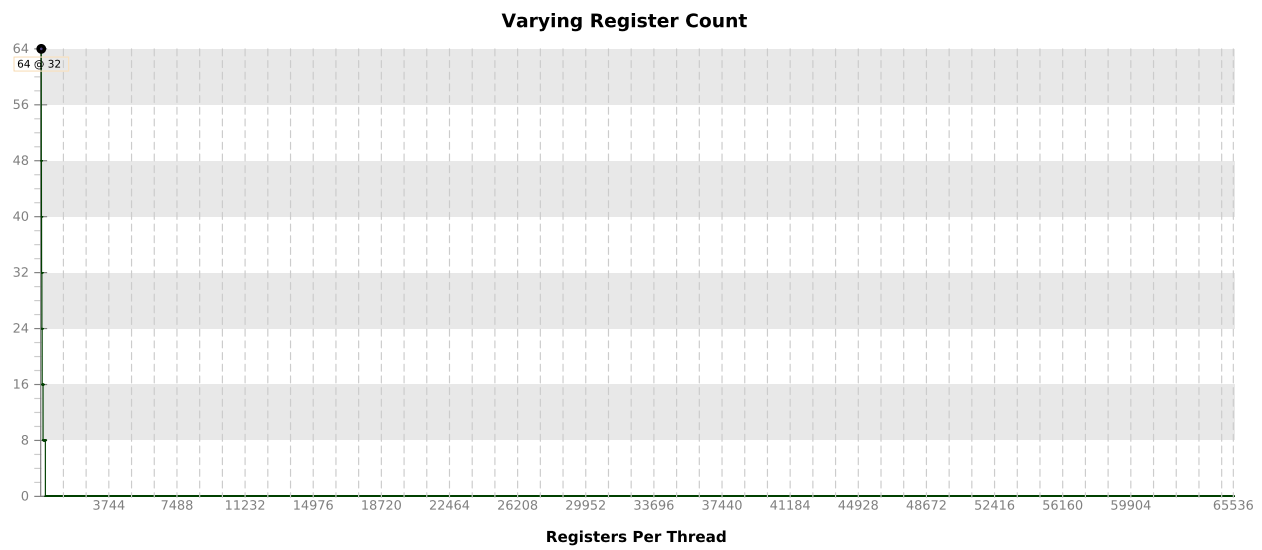
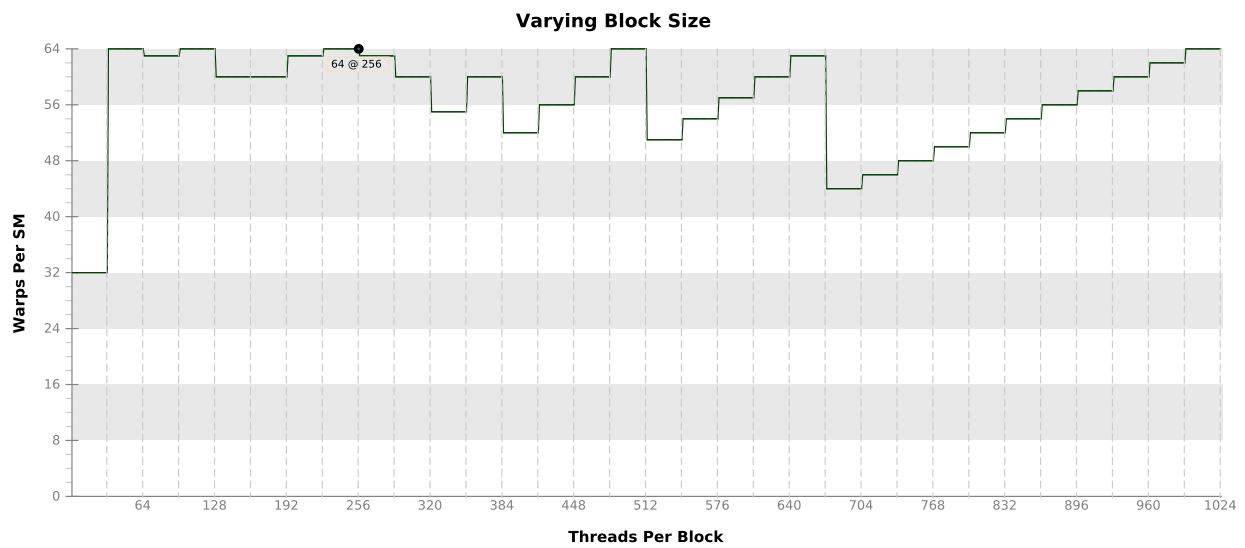
2.1. Occupancy Is Not Limiting Kernel Performance

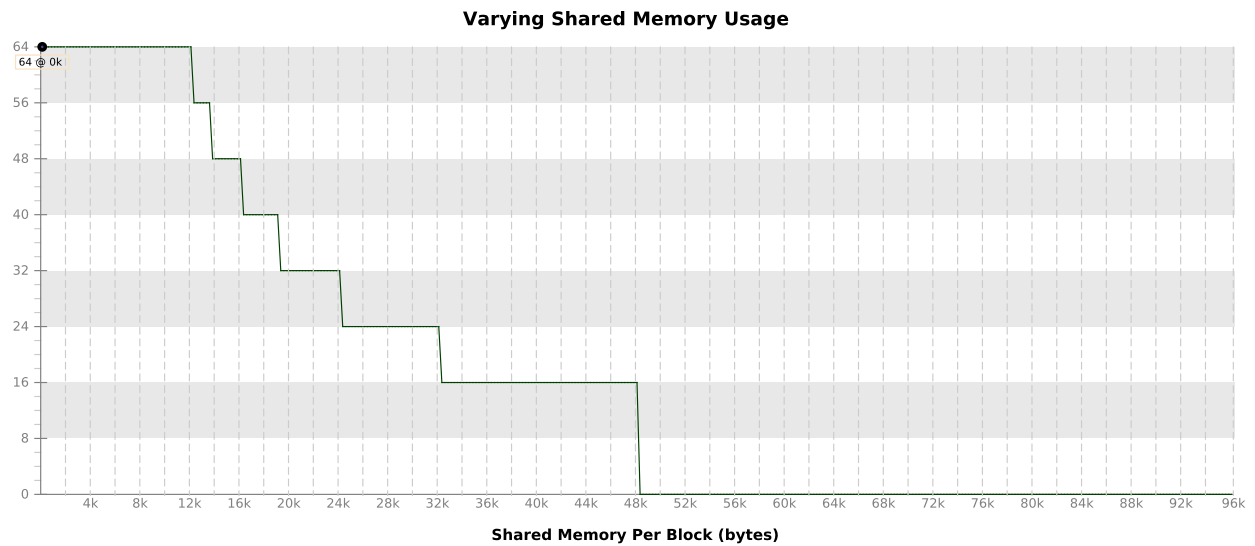
The kernel's block size, register usage, and shared memory usage allow it to fully utilize all warps on the GPU.

Variable	Achieved	Theoretical	Device Limit	Grid Size: [313,313,1] (97969 blocks) Block Size: [16,16,1] (256)
Occupancy Per SM				
Active Blocks		8	32	
Active Warps	63.46	64	64	
Active Threads		2048	2048	
Occupancy	99.2%	100%	100%	
Warps				
Threads/Block		256	1024	
Warps/Block		8	32	
Block Limit		8	32	
Registers				
Registers/Thread		32	65536	
Registers/Block		8192	65536	
Block Limit		8	32	
Shared Memory				
Shared Memory/Block		0	98304	
Block Limit		0	32	

2.2. Occupancy Charts

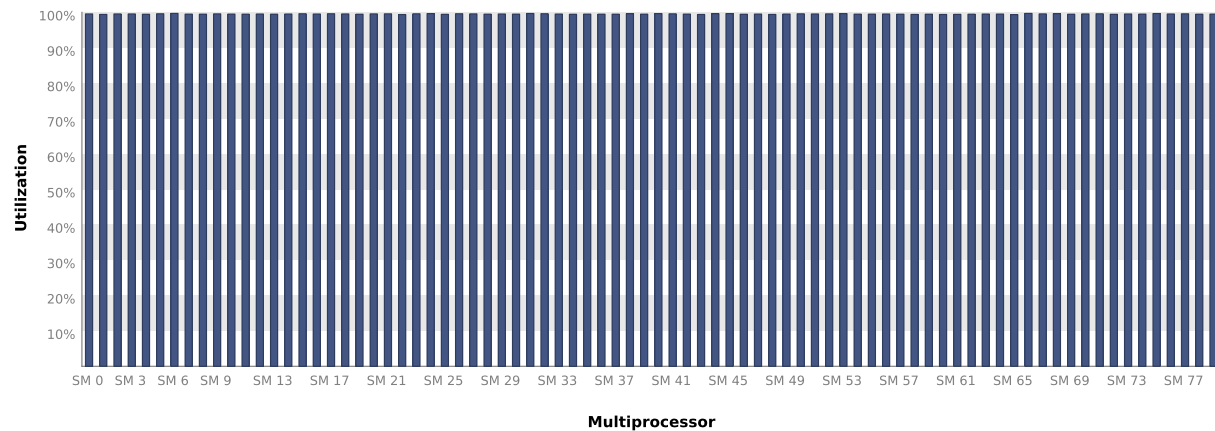
The following charts show how varying different components of the kernel will impact theoretical occupancy.





2.3. Multiprocessor Utilization

The kernel's blocks are distributed across the GPU's multiprocessors for execution. Depending on the number of blocks and the execution duration of each block some multiprocessors may be more highly utilized than others during execution of the kernel. The following chart shows the utilization of each multiprocessor during execution of the kernel.



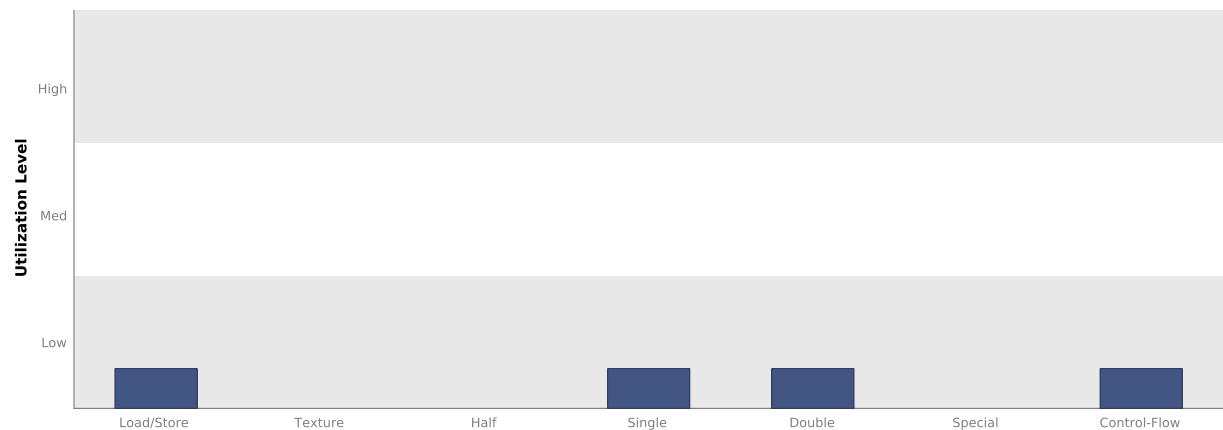
3. Compute Resources

GPU compute resources limit the performance of a kernel when those resources are insufficient or poorly utilized.

3.1. Function Unit Utilization

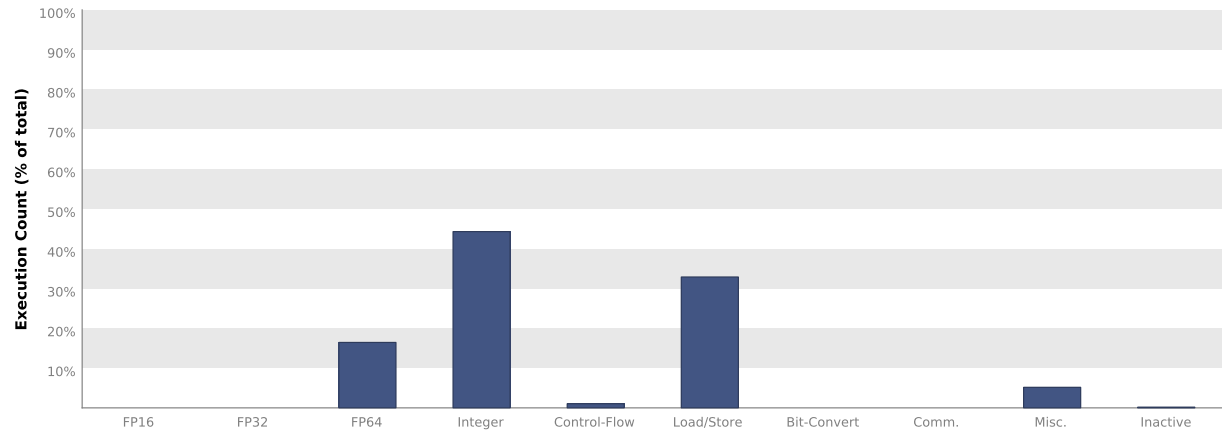
Different types of instructions are executed on different function units within each SM. Performance can be limited if a function unit is over-used by the instructions executed by the kernel. The following results show that the kernel's performance is not limited by overuse of any function unit.

- Load/Store - Load and store instructions for shared and constant memory.
- Texture - Load and store instructions for local, global, and texture memory.
- Half - Half-precision floating-point arithmetic instructions.
- 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.



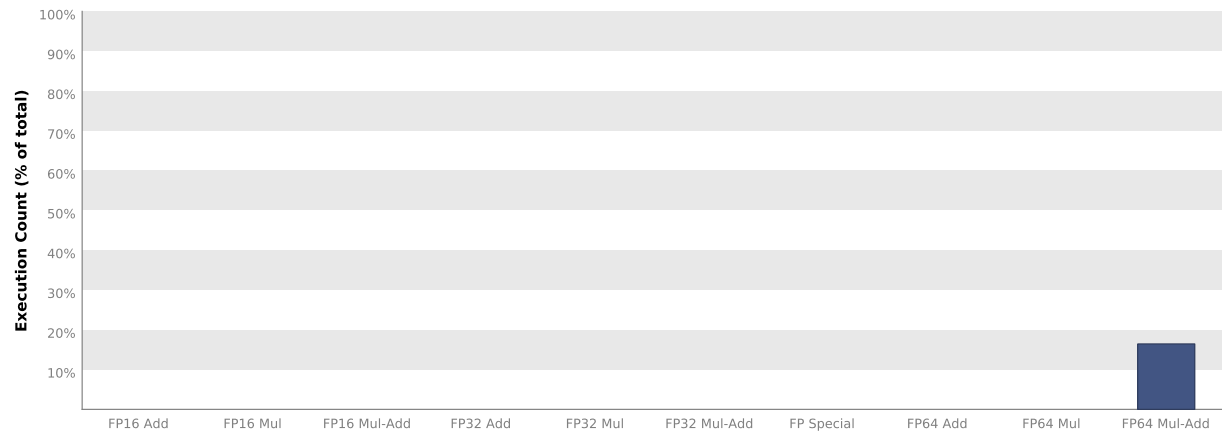
3.2. Instruction Execution Counts

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.



3.3. 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. The results below indicate that the kernel is limited by the bandwidth available to the unified cache that holds texture, global, and local data.

4.1. Global Memory Alignment and Access Pattern

Memory bandwidth is used most efficiently when each global memory load and store has proper alignment and access pattern. The analysis is per assembly instruction.

Optimization: Each entry below points to a global load or store within the kernel with an inefficient alignment or access pattern. For each load or store improve the alignment and access pattern of the memory access.

4.2. 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	0	0 B/s	
Shared Stores	0	0 B/s	
Shared Total	0	0 B/s	
L2 Cache			
Reads	4717764369	182.849 GB/s	
Writes	31225790	1.21 GB/s	
Total	4748990159	184.06 GB/s	
Unified Cache			
Local Loads	6321492	245.006 MB/s	
Local Stores	9958760	385.978 MB/s	
Global Loads	66154091965	2,563.978 GB/s	
Global Stores	18313138	709.774 MB/s	
Texture Reads	59365174973	9,203.42 GB/s	
Unified Total	125553860328	11,768.739 GB/s	
Device Memory			
Reads	2496711727	96.767 GB/s	
Writes	23649922	916.616 MB/s	
Total	2520361649	97.683 GB/s	
System Memory			
[PCIe configuration: Gen3 x16, 8 Gbit/s]			
Reads	72151	2.796 MB/s	
Writes	71139	2.757 MB/s	