Analysis Report

sharedDgemm(int, int, int, double, double const *, double const *, double, double*)

Duration	232.19868 ms (232,198,681 ns)
Grid Size	[256,512,1]
Block Size	[16,16,1]
Registers/Thread	30
Shared Memory/Block	4 KiB
Shared Memory Executed	32 KiB
Shared Memory Bank Size	4 B

[0] Tesla P100-PCIE-16GB

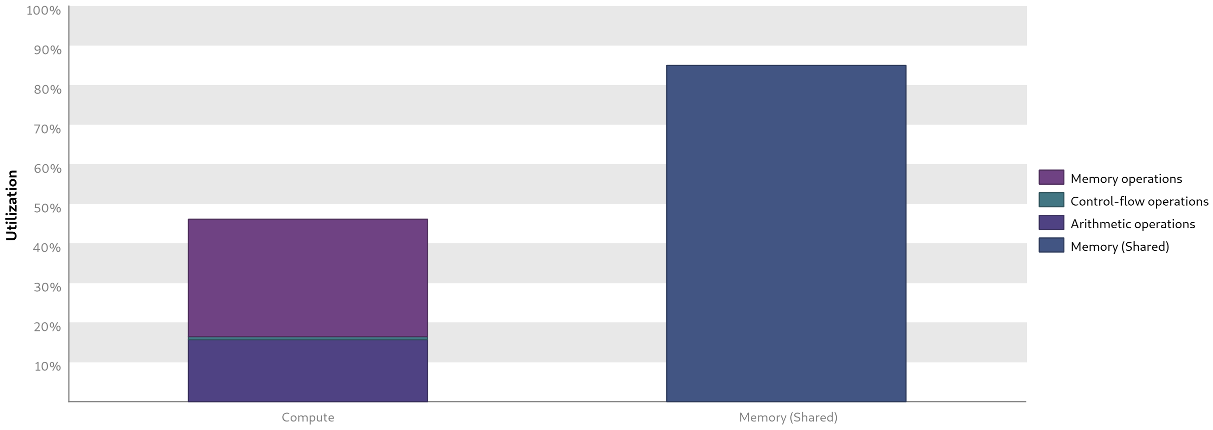
GPU UUID	GPU-200a532d-8549-0ea2-b739-7716866a30fd
Compute Capability	6.0
Max. Threads per Block	1024
Max. Threads per Multiprocessor	2048
Max. Shared Memory per Block	48 KiB
Max. Shared Memory per Multiprocessor	64 KiB
Max. Registers per Block	65536
Max. Registers per Multiprocessor	65536
Max. Grid Dimensions	[2147483647, 65535, 65535]
Max. Block Dimensions	[1024, 1024, 64]
Max. Warps per Multiprocessor	64
Max. Blocks per Multiprocessor	32
Half Precision FLOP/s	9.523 TeraFLOP/s
Single Precision FLOP/s	9.523 TeraFLOP/s
Double Precision FLOP/s	4.761 TeraFLOP/s
Number of Multiprocessors	56
Multiprocessor Clock Rate	1.329 GHz
Concurrent Kernel	true
Max IPC	3
Threads per Warp	32
Global Memory Bandwidth	732.16 GB/s
Global Memory Size	15.899 GiB
Constant Memory Size	64 KiB
L2 Cache Size	4 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 "sharedDgemm" 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 "Tesla P100-PCIE-16GB" 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.



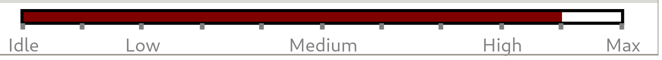
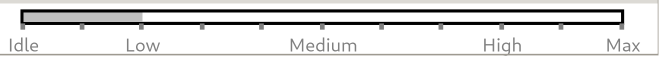
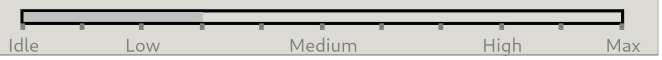
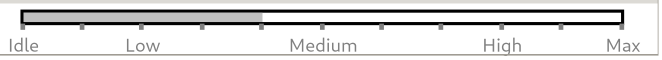
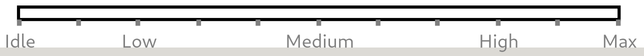
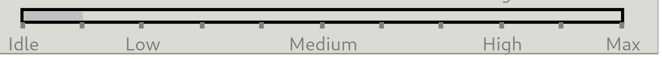
2. Memory Bandwidth

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	12884901888	7,111.755 GB/s	
Shared Stores	1073741824	592.646 GB/s	
Shared Total	13958643712	7,704.402 GB/s	
L2 Cache			
Reads	3095269244	427.104 GB/s	
Writes	33555050	4.63 GB/s	
Total	3128824294	431.735 GB/s	
Unified Cache			
Local Loads	0	0 B/s	
Local Stores	0	0 B/s	
Global Loads	17213423618	1,187.608 GB/s	
Global Stores	33554432	4.63 GB/s	
Texture Reads	4303355904	593.804 GB/s	
Unified Total	21550333954	1,786.041 GB/s	
Device Memory			
Reads	2054531333	283.497 GB/s	
Writes	8435099	1.164 GB/s	
Total	2062966432	284.661 GB/s	
System Memory			
[PCIe configuration: Gen3 x16, 8 Gbit/s]			
Reads	0	0 B/s	
Writes	5	689 B/s	

2.2. Memory Statistics

The following chart shows a summary view of the memory hierarchy of the CUDA programming model. The green nodes in the diagram depict logical memory space whereas blue nodes depicts actual hardware unit on the chip. For the various caches the reported percentage number states the cache hit rate; that is the ratio of requests that could be served with data locally available to the cache over all requests made.

The links between the nodes in the diagram depict the data paths between the SMs to the memory spaces into the memory system. Different metrics are shown per data path. The data paths from the SMs to the memory spaces report the total number of memory instructions executed, it includes both read and write operations. The data path between memory spaces and "Unified Cache" or "Shared Memory" reports the total amount of memory requests made (read or write). All other data paths report the total amount of transferred memory in bytes.

3. Instruction and Memory Latency

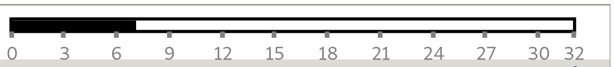
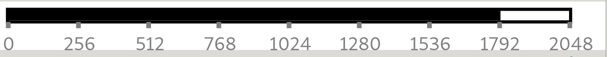
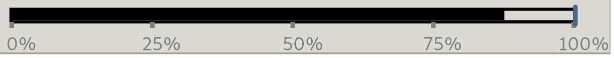
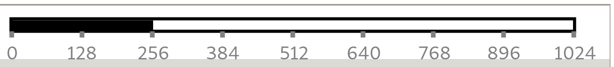
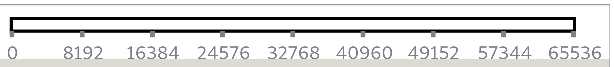
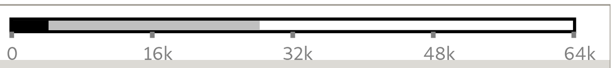
Instruction and memory latency limit the performance of a kernel when the GPU does not have enough work to keep busy. The performance of latency-limited kernels can often be improved by increasing occupancy. Occupancy is a measure of how many warps the kernel has active on the GPU, relative to the maximum 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.

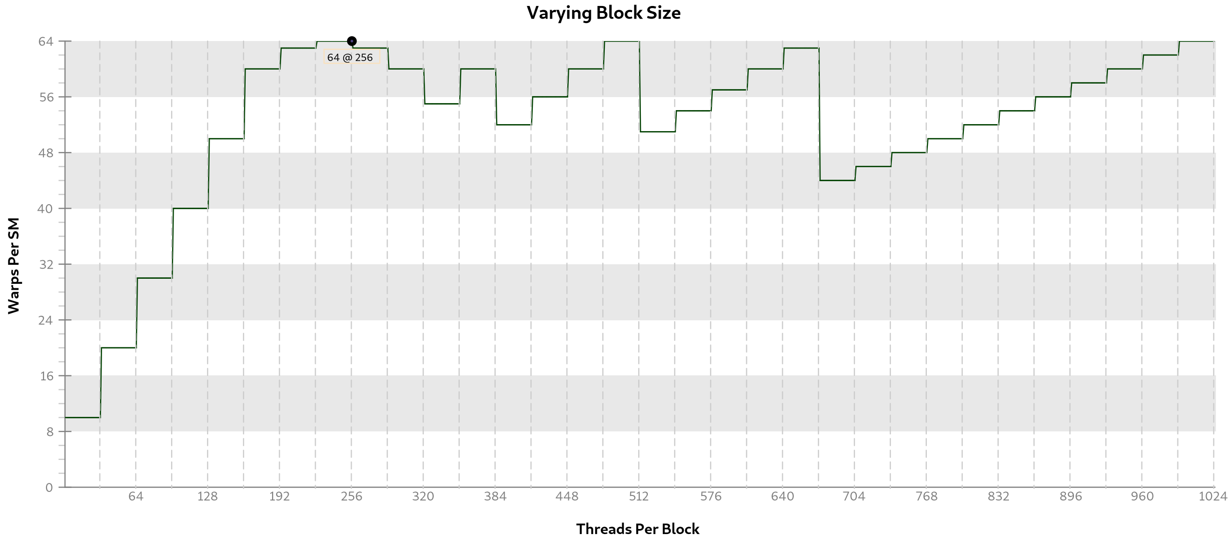
The kernel uses 4 KiB of shared memory for each block. This shared memory usage is likely preventing the kernel from fully utilizing the GPU. Device "Tesla P100-PCIE-16GB" is configured to have 64 KiB of shared memory for each SM. Because the kernel uses 4 KiB of shared memory for each block each SM is limited to simultaneously executing 8 blocks (64 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.

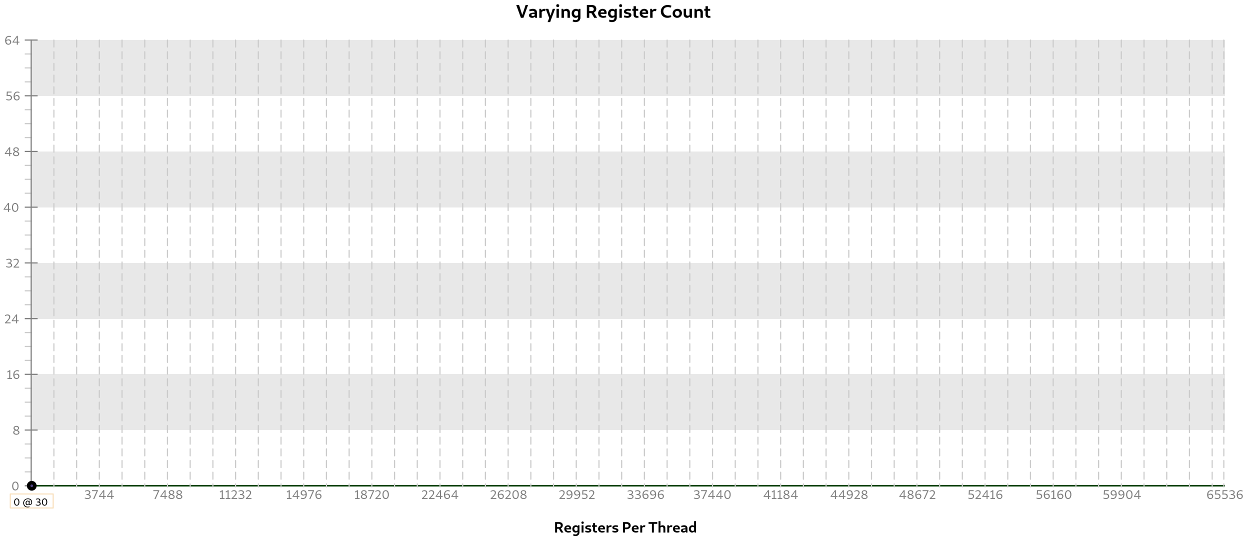
Optimization: Reduce shared memory usage to increase the number of blocks that can execute on each SM. You can also increase the number of blocks that can execute on each SM by increasing the amount of shared memory available to your kernel. You do this by setting the preferred cache configuration to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

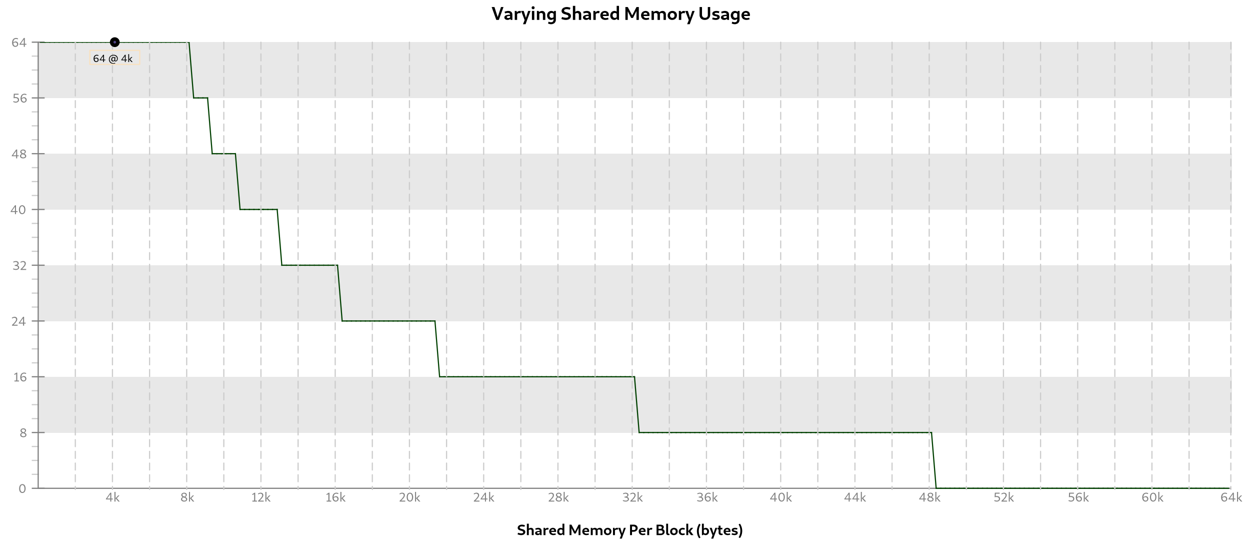
Variable	Achieved	Theoretical	Device Limit	Grid Size: [256,512,1] (131072 blocks) Block Size: [16,16,1] (256 threads)
Occupancy Per SM				
Active Blocks		7	32	
Active Warps	63.95	56	64	
Active Threads		1792	2048	
Occupancy	99.9%	87.5%	100%	
Warps				
Threads/Block		256	1024	
Warps/Block		8	32	
Block Limit		8	32	
Registers				
Registers/Thread		30	65536	
Registers/Block		8192	65536	
Block Limit		8	32	
Shared Memory				
Shared Memory/Block		4096	65536	
Block Limit		7	32	

3.2. Occupancy Charts

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

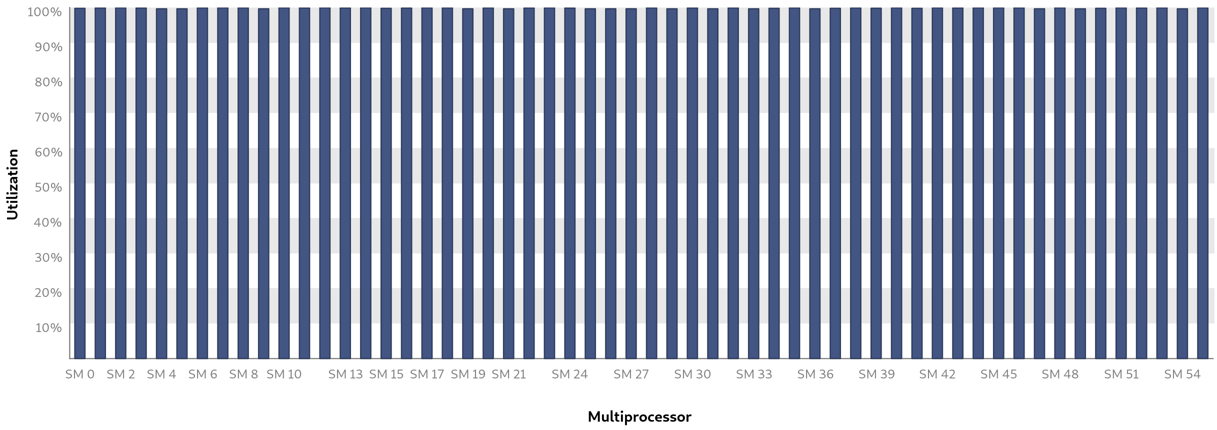

Varying Block Size


Varying Register Count



3.3. 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.



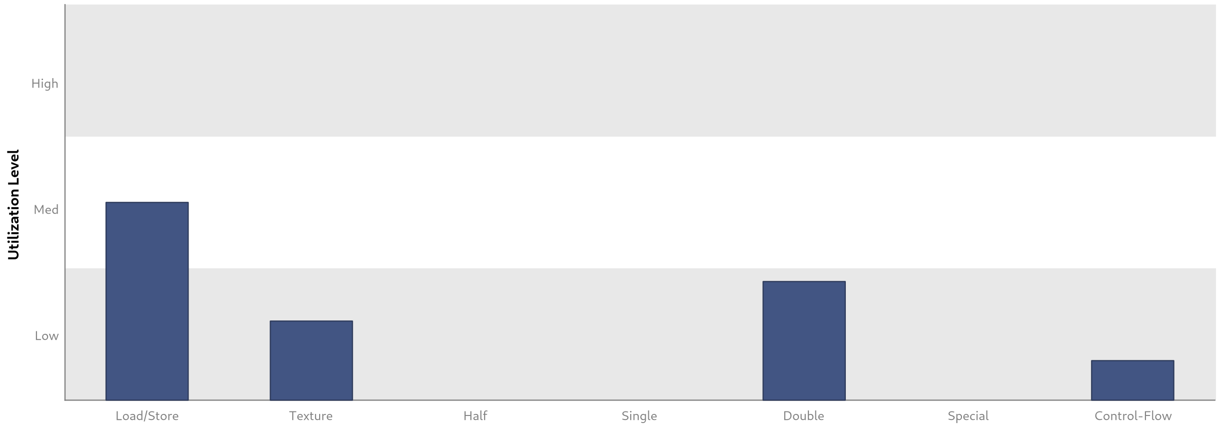
4. Compute Resources

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

4.1. Function Unit Utilization

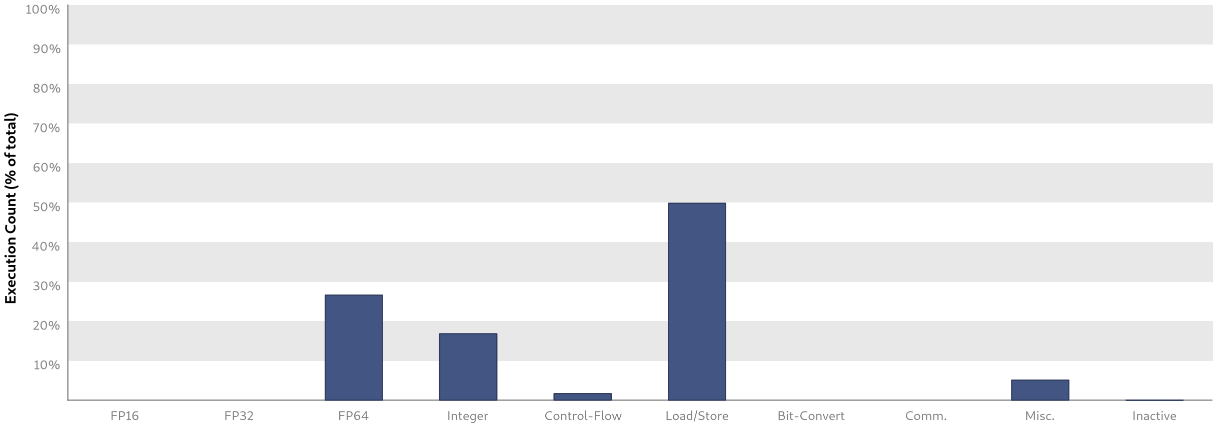
Different types of instructions are executed on different function units within each SM. Performance can be limited if a function unit is over-used by the instructions executed by the kernel. The following results show that the kernel's performance is not limited by overuse of any function unit.

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.



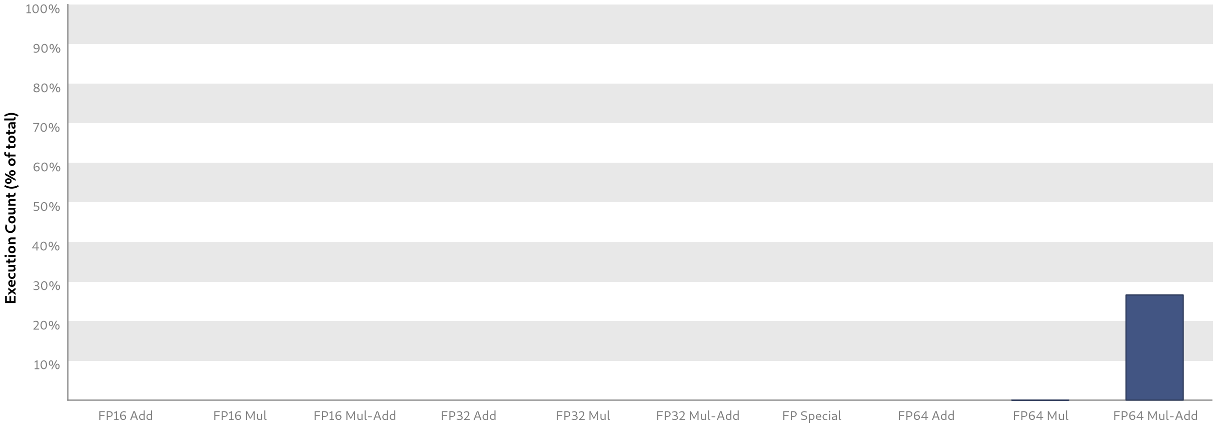
4.2. Instruction Execution Counts

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.



4.3. 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.


Execution Count (% of total)
FP16 Add
FP16 Mul
FP16 Mul-Add
FP32 Add
FP32 Mul
FP32 Mul-Add
FP Special
FP64 Add
FP64 Mul
FP64 Mul-Add