# Analysis Report

## gen_kmul4_sgemmNT2_core(float const *, int, float const *, int, float*, int, int, int, int, float const *, float const *, float, float, int)

| | |
|---|---|
| Duration | 10.464 µs |
| Grid Size | [ 1,49,1 ] |
| Block Size | [ 16,4,1 ] |
| Registers/Thread | 84 |
| Shared  Memory/Block | 256 B |
| Shared Memory Requested | 96 KiB |
| Shared Memory Executed | 96 KiB |
| Shared Memory Bank Size | 4 B |

[0] GeForce GTX 1080

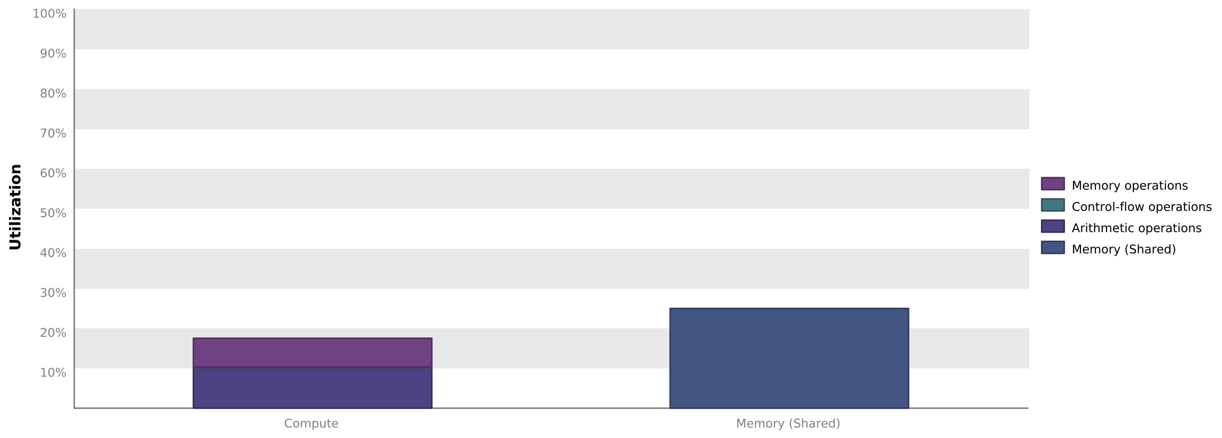
| | |
|---|---|
| GPU UUID | GPU-edd19385-a5f1-ce46-e1d9-61408827cccc |
| Compute Capability | 6.1 |
| Max. Threads per Block | 1024 |
| Max. Threads per Multiprocessor | 2048 |
| Max. Shared Memory per Block | 48 KiB |
| Max. Shared Memory per Multiprocessor | 96 KiB |
| Max. Registers per Block | 65536 |
| 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 | 72.9 GigaFLOP/s |
| Single Precision FLOP/s | 9.331 TeraFLOP/s |
| Double Precision FLOP/s | 291.6 GigaFLOP/s |
| Number of Multiprocessors | 20 |
| Multiprocessor Clock Rate | 1.823 GHz |
| Concurrent Kernel | true |
| Max IPC | 6 |
| Threads per Warp | 32 |
| Global Memory Bandwidth | 320.32 GB/s |
| Global Memory Size | 7.923 GiB |
| Constant Memory Size | 64 KiB |
| L2 Cache Size | 2 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 "gen_kmul4_sgemmNT2_core" 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 1080". 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 results below indicate that the GPU does not have enough work because the kernel does not execute enough blocks.

### 2.1. Grid Size Too Small To Hide Compute And Memory Latency

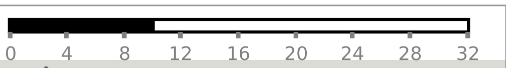
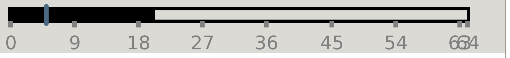
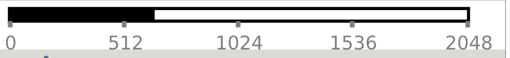
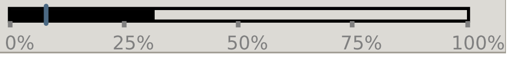
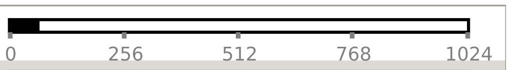
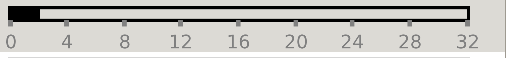
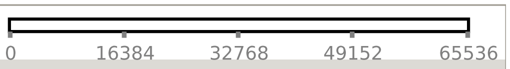
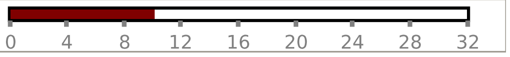
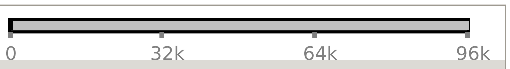
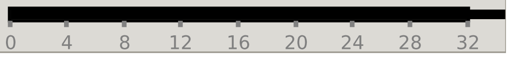
The kernel does not execute enough blocks to hide memory and operation latency. Typically the kernel grid size must be large enough to fill the GPU with multiple "waves" of blocks. Based on theoretical occupancy, device "GeForce GTX 1080" can simultaneously execute 10 blocks on each of the 20 SMs, so the kernel may need to execute a multiple of 200 blocks to hide the compute and memory latency. If the kernel is executing concurrently with other kernels then fewer blocks will be required because the kernel is sharing the SMs with those kernels.

*Optimization: Increase the number of blocks executed by the kernel.*

### 2.2. GPU Utilization Is Limited By Register Usage

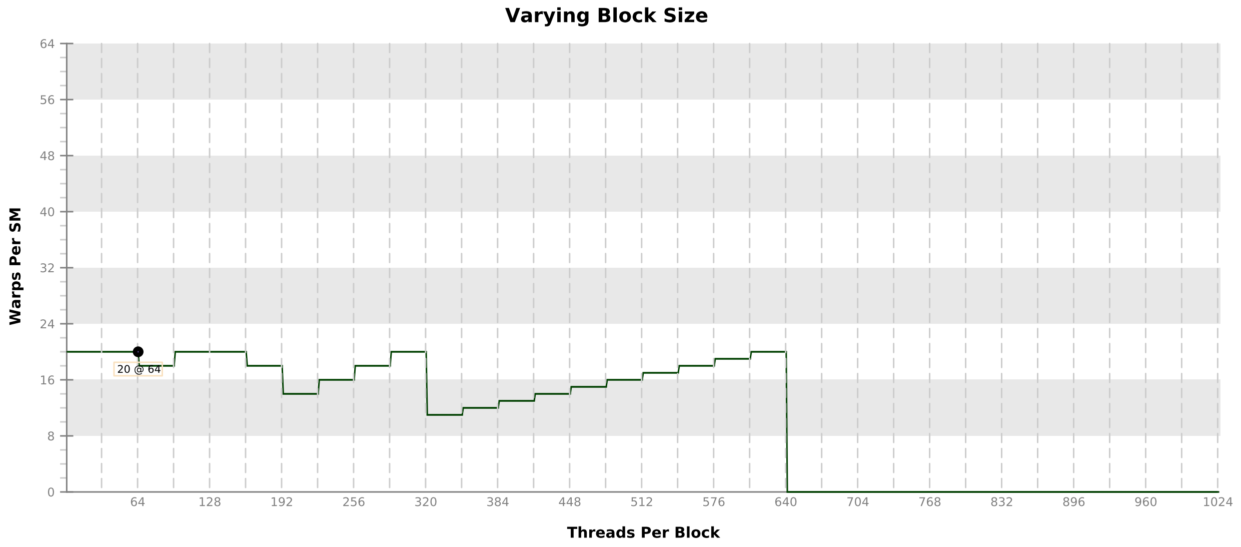
The kernel uses 84 registers for each thread (5376 registers for each block). This register usage is likely preventing the kernel from fully utilizing the GPU. Device "GeForce GTX 1080" provides up to 65536 registers for each block. Because the kernel uses 5376 registers for each block each SM is limited to simultaneously executing 10 blocks (20 warps). Chart "Varying Register Count" below shows how changing register usage will change the number of blocks that can execute on each SM.

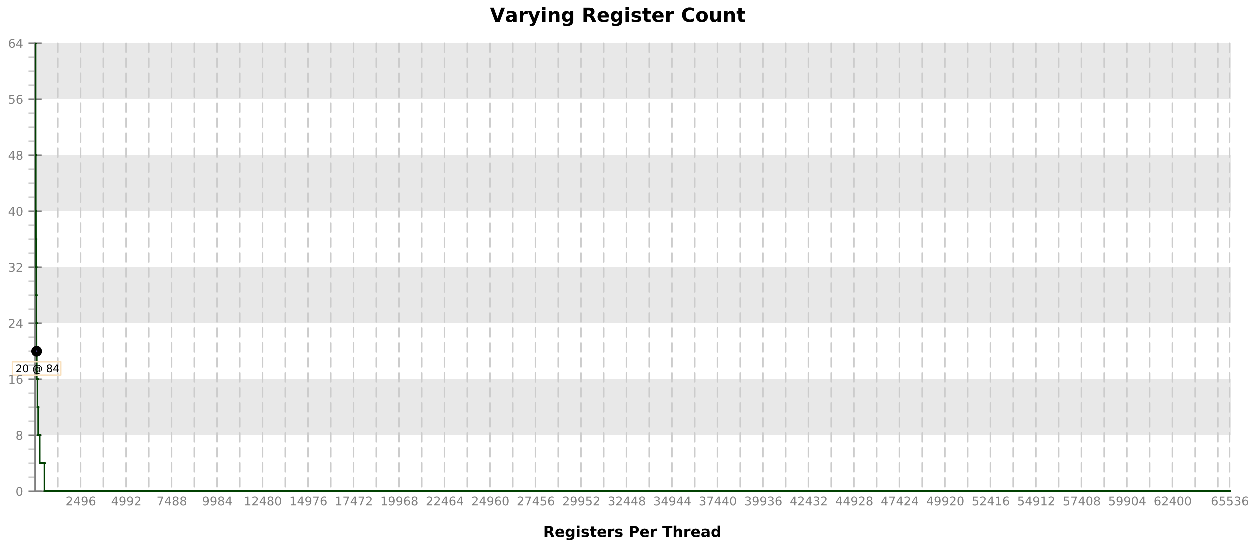
*Optimization: Use the -maxrregcount flag or the __launch_bounds__ qualifier to decrease the number of registers used by each thread. This will increase the number of blocks that can execute on each SM. On devices with Compute Capability 5.2 turning global cache off can increase the occupancy limited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

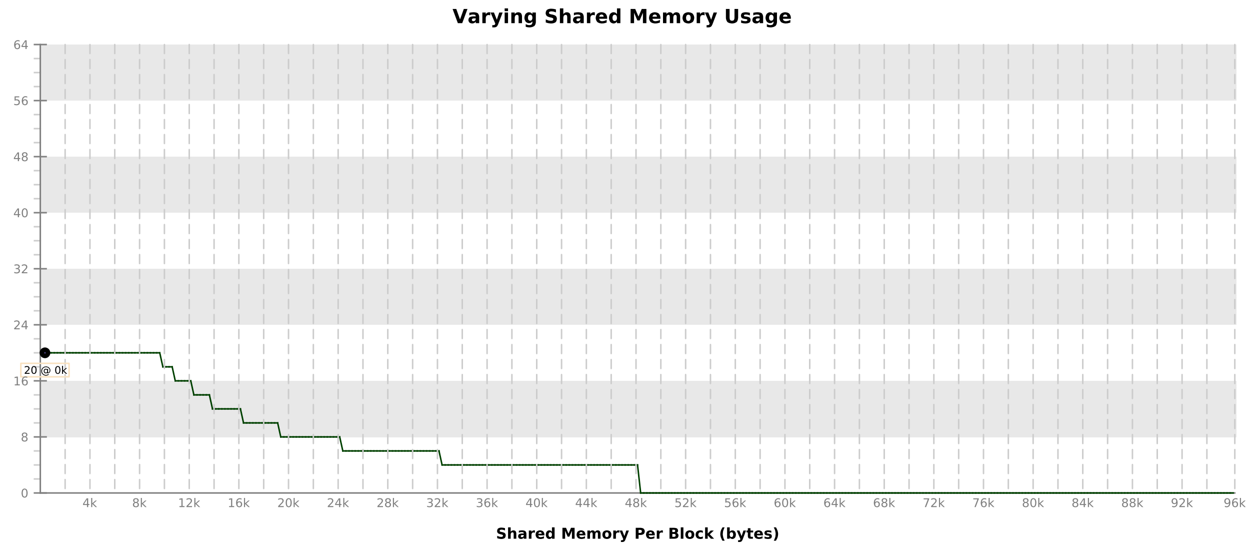
| Variable | Achieved | Theoretical | Device Limit | Grid Size: [ 1,49,1 ] (49 blocks) Block Size: [ 16,4,1 ] (64 |
|---|---|---|---|---|
| Occupancy Per SM | | | | |
| Active Blocks | | 10 | 32 |  |
| Active Warps | 4.66 | 20 | 64 |  |
| Active Threads | | 640 | 2048 |  |
| Occupancy | 7.3% | 31.2% | 100% |  |
| Warps | | | | |
| Threads/Block | | 64 | 1024 |  |
| Warps/Block | | 2 | 32 |  |
| Block Limit | | 32 | 32 | |
| Registers | | | | |
| Registers/Thread | | 84 | 65536 |  |
| Registers/Block | | 5632 | 65536 | |
| Block Limit | | 10 | 32 |  |
| Shared Memory | | | | |
| Shared Memory/Block | | 256 | 98304 |  |
| Block Limit | | 384 | 32 |  |

## 2.3. Occupancy Charts

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







Varying Shared Memory Usage
64
56
48
40
32
24
16
8
0
20 @ 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)

# 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. Divergent Branches

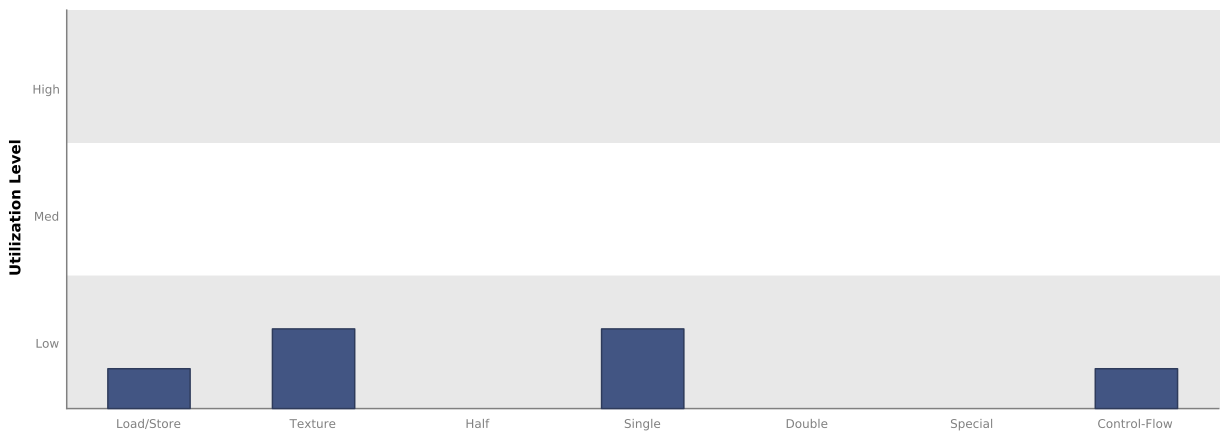
Compute resource are used most efficiently when all threads in a warp have the same branching behavior. When this does not occur the branch is said to be divergent. Divergent branches lower warp execution efficiency which leads to inefficient use of the GPU's compute resources.

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

## 3.2. Function Unit Utilization

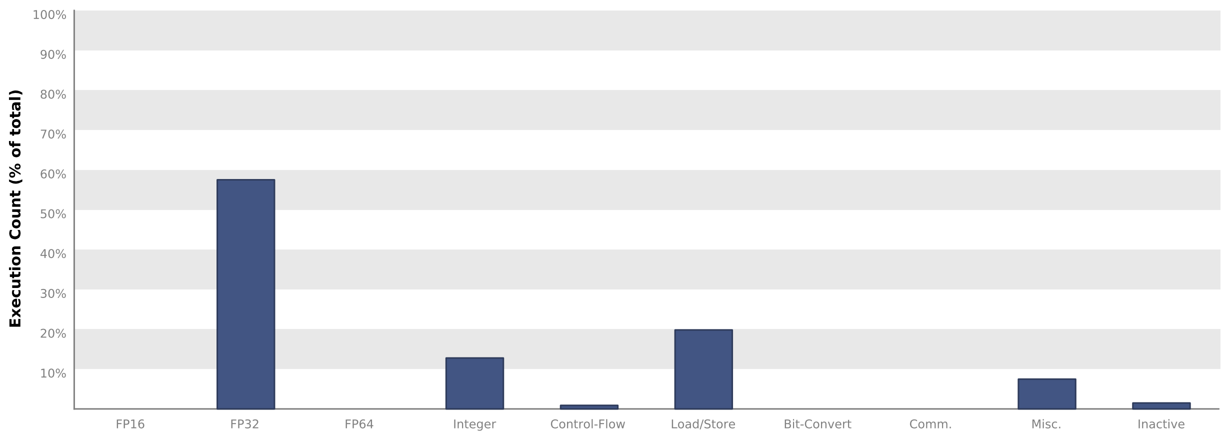
Different types of instructions are executed on different function units within each SM. Performance can be limited if a function unit is over-used by the instructions executed by the kernel. The following results show that the kernel's performance is not limited by overuse of any function unit.

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.



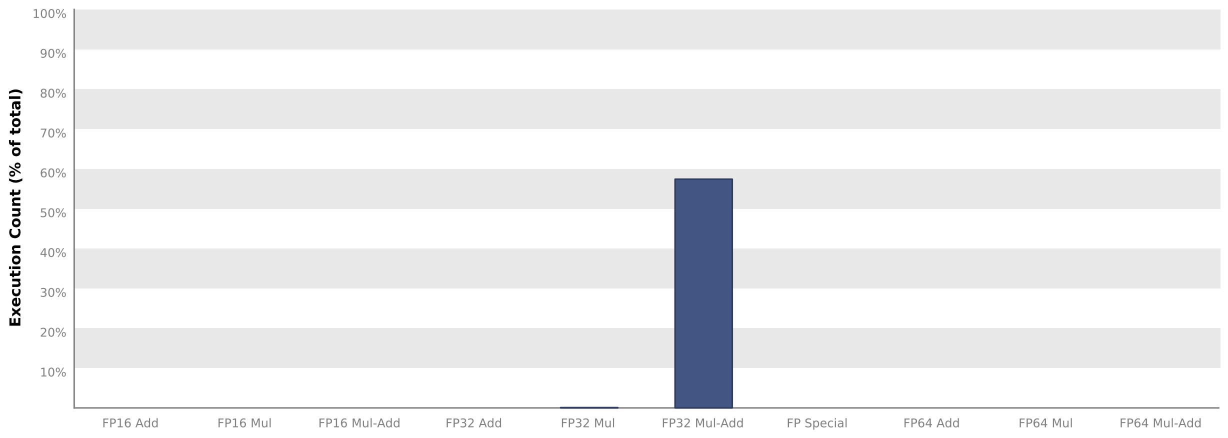
## 3.3. Instruction Execution Counts

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.



### 3.4. 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.

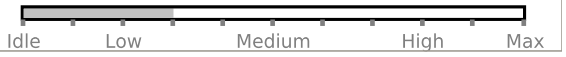
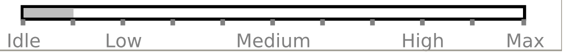
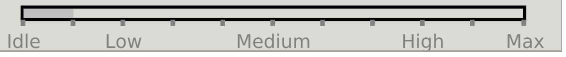
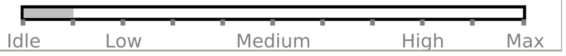


# 4. Memory Bandwidth

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 | |
|---|---|---|---|
| Shared Memory | | | |
| Shared Loads | 78400 | 1,047.953 GB/s | |
| Shared Stores | 2450 | 32.749 GB/s | |
| Shared Total | 80850 | 1,080.702 GB/s |  |
| L2 Cache | | | |
| Reads | 25172 | 84.117 GB/s | |
| Writes | 1581 | 5.283 GB/s | |
| Total | 26753 | 89.4 GB/s |  |
| Unified Cache | | | |
| Local Loads | 0 | 0 B/s | |
| Local Stores | 0 | 0 B/s | |
| Global Loads | 67375 | 81.871 GB/s | |
| Global Stores | 1568 | 5.24 GB/s | |
| Texture Reads | 49000 | 163.743 GB/s | |
| Unified Total | 117943 | 250.854 GB/s |  |
| Device Memory | | | |
| Reads | 4 | 13.367 MB/s | |
| Writes | 1068 | 3.569 GB/s | |
| Total | 1072 | 3.582 GB/s |  |
| System Memory | | | |
| [ PCIe configuration: Gen3 x16, 8 Gbit/s ] | | | |
| Reads | 0 | 0 B/s | Idle Low Medium High Max |
| Writes | 5 | 16.708 MB/s | Idle Low Medium High Max |