Analysis Report

findMatch(unsigned char*, unsigned long, int, unsigned char*, unsigned long, int, int, unsigned int*)

Duration	8.213 ms (8,212,994 ns)
Grid Size	[14337,1,1]
Block Size	[1024,1,1]
Registers/Thread	20
Shared Memory/Block	0 B
Shared Memory Requested	48 KiB
Shared Memory Executed	48 KiB
Shared Memory Bank Size	4 B

[0] Tesla K20c

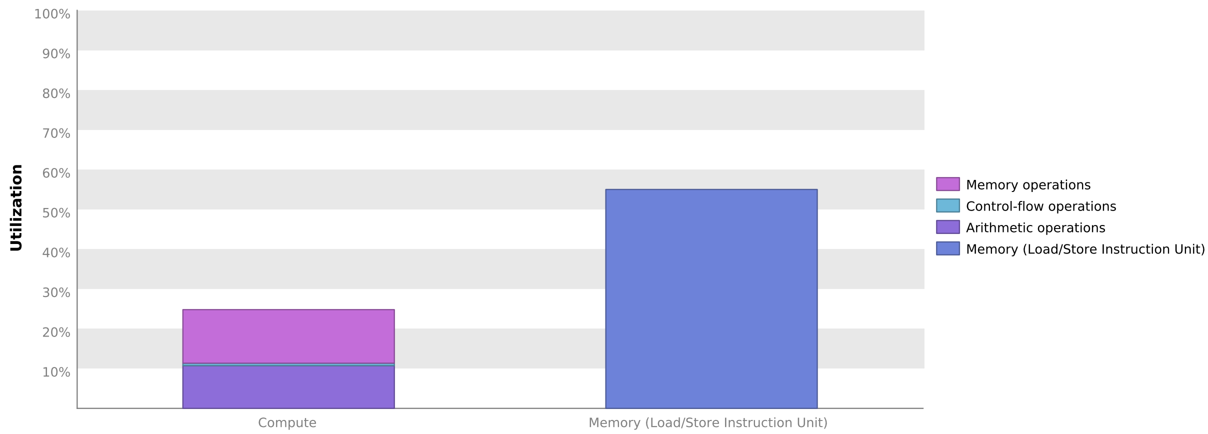
GPU UUID	GPU-8755f09f-ebc6-0c4c-349a-22c22f181d94
Compute Capability	3.5
Max. Threads per Block	1024
Max. Shared Memory per Block	48 KiB
Max. Registers per Block	65536
Max. Grid Dimensions	[2147483647, 65535, 65535]
Max. Block Dimensions	[1024, 1024, 64]
Max. Warps per Multiprocessor	64
Max. Blocks per Multiprocessor	16
Single Precision FLOP/s	3.522 TeraFLOP/s
Double Precision FLOP/s	1.174 TeraFLOP/s
Number of Multiprocessors	13
Multiprocessor Clock Rate	705.5 MHz
Concurrent Kernel	true
Max IPC	7
Threads per Warp	32
Global Memory Bandwidth	208 GB/s
Global Memory Size	4.687 GiB
Constant Memory Size	64 KiB
L2 Cache Size	1.25 MiB
Memcpy Engines	2
PCIe Generation	2
PCIe Link Rate	5 Gbit/s
PCIe Link Width	8

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 "findMatch" 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 K20c". 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.



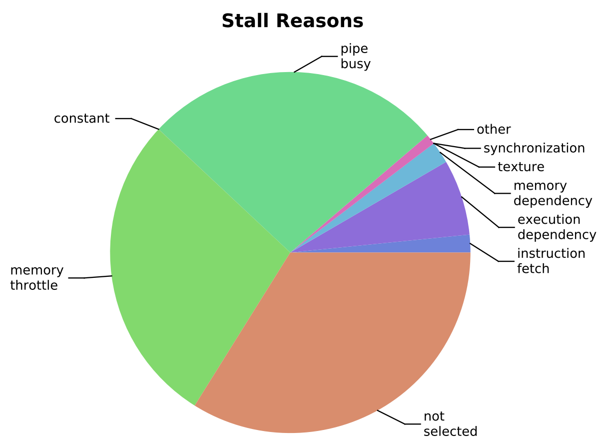
2. Instruction and Memory Latency

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 instruction execution is stalling excessively.

2.1. Instruction Latencies May Be Limiting Performance

Instruction stall reasons indicate the condition that prevents warps from executing on any given cycle. The following chart shows the break-down of stalls reasons averaged over the entire execution of the kernel. The kernel has good theoretical and achieved occupancy indicating that there are likely sufficient warps executing on each SM. Since occupancy is not an issue it is likely that performance is limited by the instruction stall reasons described below.
Memory Throttle - Large number of pending memory operations prevent further forward progress. These can be reduced by combining several memory transactions into one.
Texture - The texture sub-system is fully utilized or has too many outstanding requests.
Synchronization - The warp is blocked at a __syncthreads() call.
Instruction Fetch - The next assembly instruction has not yet been fetched.
Memory Dependency - A load/store cannot be made because the required resources are not available or are fully utilized, or too many requests of a given type are outstanding. Data request stalls can potentially be reduced by optimizing memory alignment and access patterns.
Execution Dependency - An input required by the instruction is not yet available. Execution dependency stalls can potentially be reduced by increasing instruction-level parallelism.
Pipeline Busy - The compute resource(s) required by the instruction is not yet available.
Constant - A constant load is blocked due to a miss in the constants cache.
Not Selected - Warp was ready to issue, but some other warp issued instead. You may be able to sacrifice occupancy without impacting latency hiding and doing so may help improve cache hit rates.

Optimization: Resolve the primary stall issue; not selected.



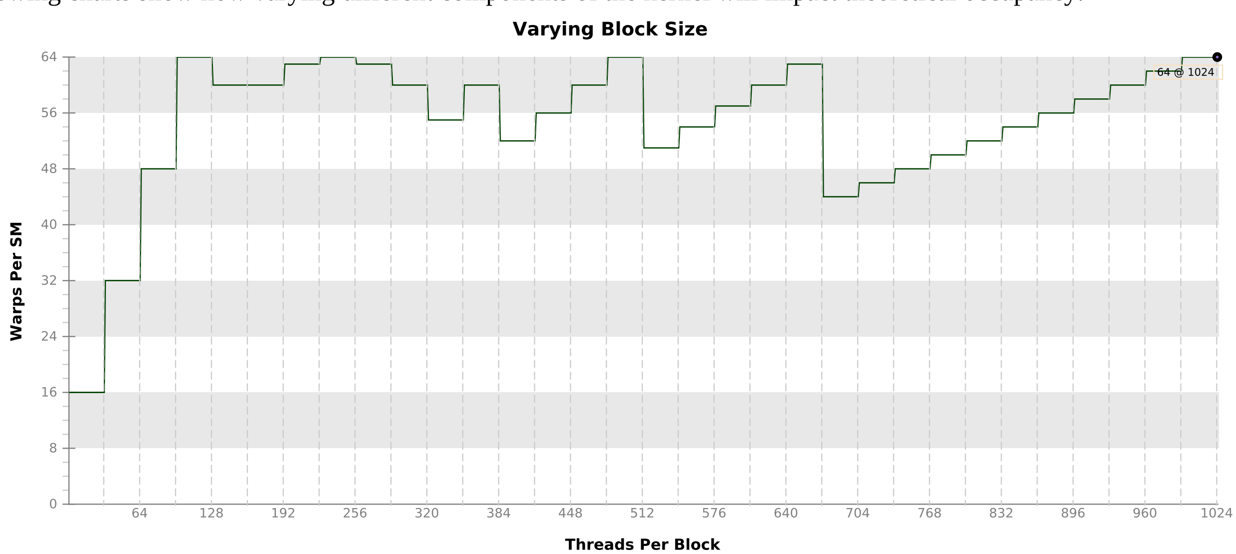
2.2. Occupancy Is Not Limiting Kernel Performance

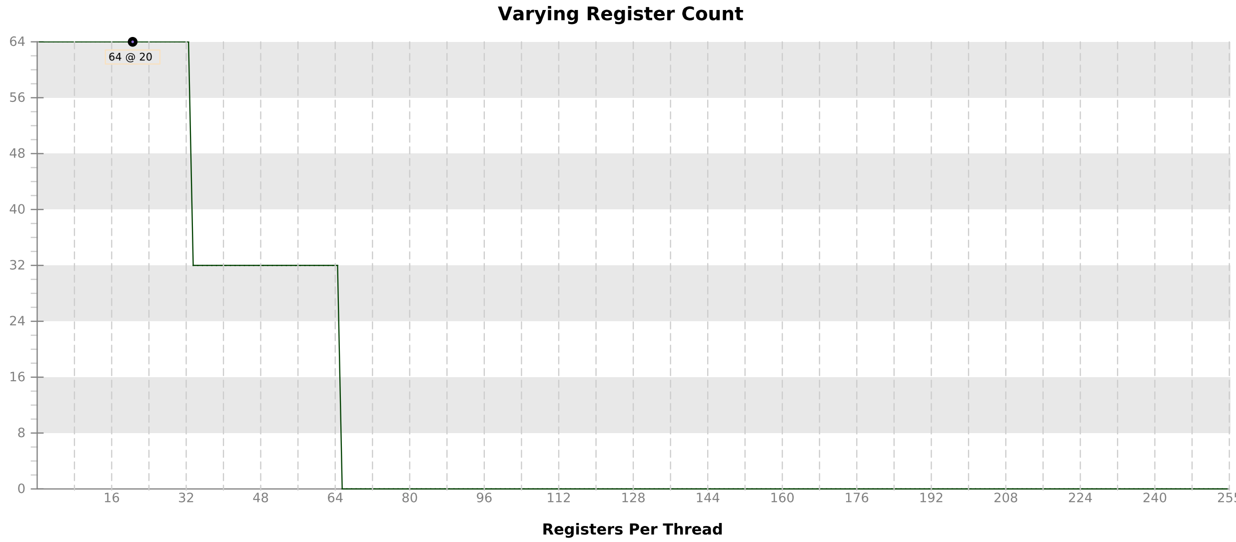
The kernel's block size, register usage, and shared memory usage allow it to fully utilize all warps on the GPU.

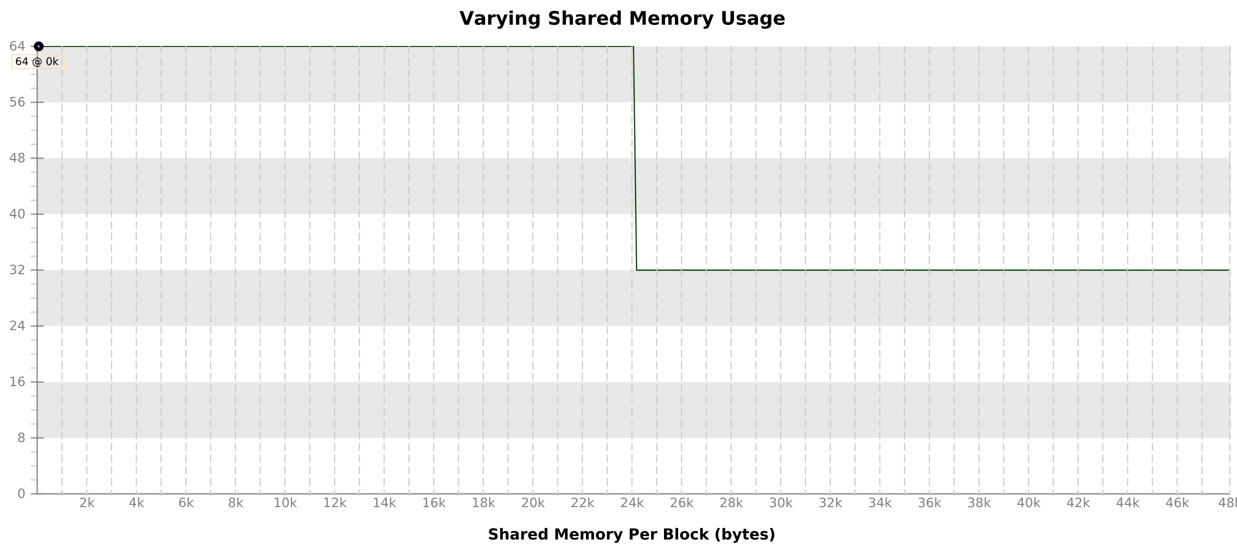
Variable	Achieved	Theoretical	Device Limit	Grid Size: [14337,1,1] (14337 blocks) Block Size: [1024,1,1] (10:
Occupancy Per SM				
Active Blocks		2	16	
Active Warps	49.8	64	64	
Active Threads		2048	2048	
Occupancy	77.8%	100%	100%	
Warps				
Threads/Block		1024	1024	
Warps/Block		32	32	
Block Limit		2	16	
Registers				
Registers/Thread		20	255	
Registers/Block		24576	65536	
Block Limit		2	16	
Shared Memory				
Shared Memory/Block		0	49152	
Block Limit			16	

2.3. Occupancy Charts

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




Varying Register Count
64 @ 20
Registers Per Thread


Varying Shared Memory Usage
64 @ 0k
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.

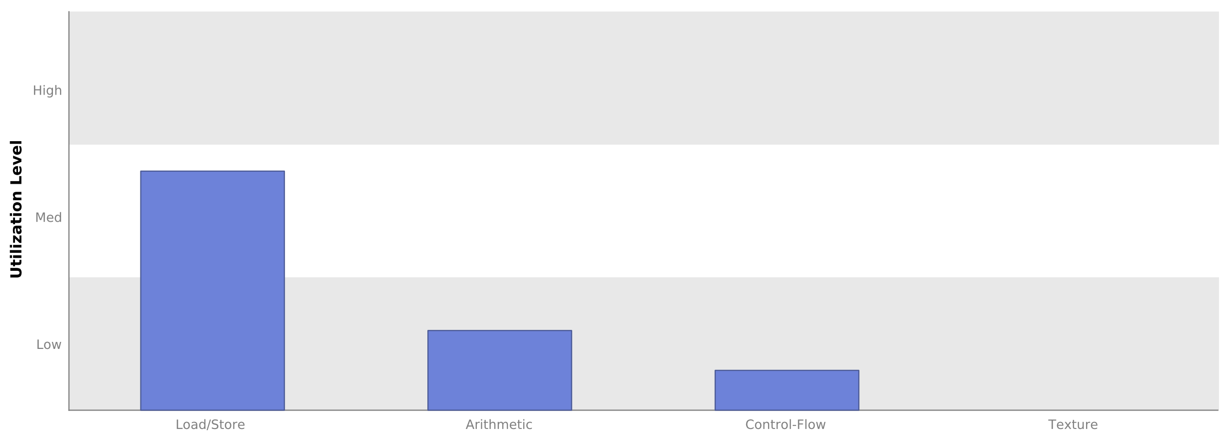
3.1. Divergent Branches

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.

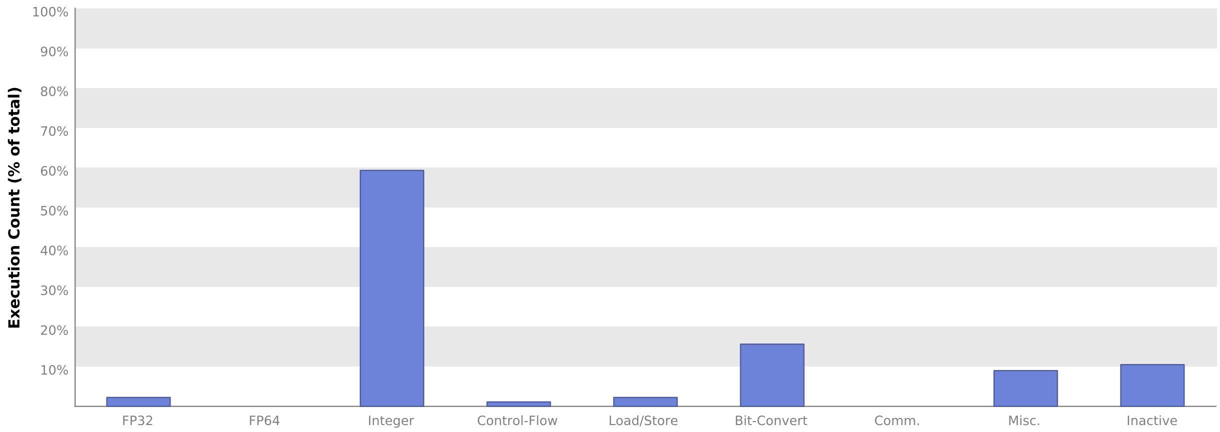
3.2. Function Unit Utilization

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 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.



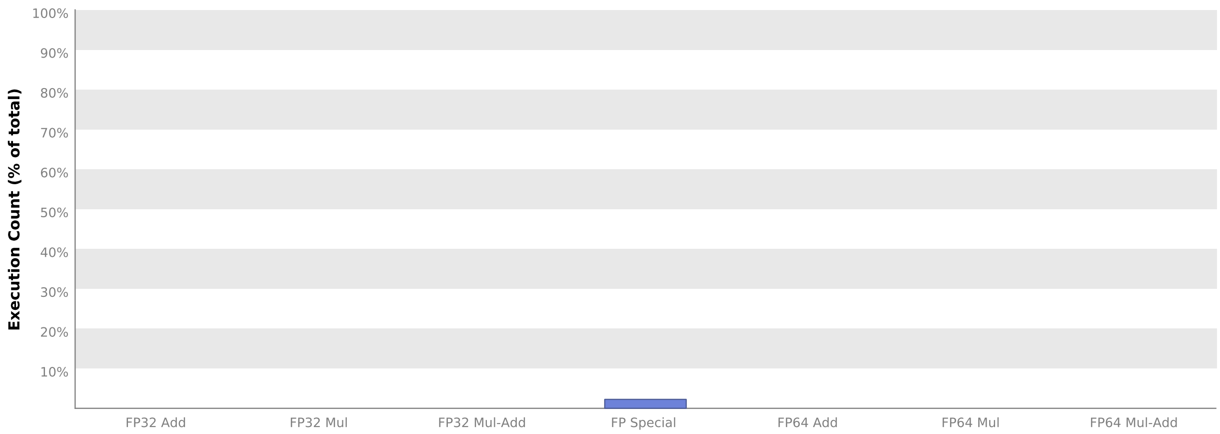
3.3. Instruction Execution Counts

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	
L1/Shared Memory			
Local Loads	0	0 B/s	
Local Stores	0	0 B/s	
Shared Loads	0	0 B/s	
Shared Stores	0	0 B/s	
Global Loads	29363584	116.592 GB/s	
Global Stores	2	7.941 kB/s	
Atomic	0	0 B/s	
L1/Shared Total	29363586	116.592 GB/s	Idle Low Medium High Max
L2 Cache			
L1 Reads	29363584	116.592 GB/s	
L1 Writes	2	7.941 kB/s	
Texture Reads	0	0 B/s	
Noncoherent Reads	0	0 B/s	
Atomic	0	0 B/s	
Total	29363586	116.592 GB/s	Idle Low Medium High Max
Texture Cache			
Reads	0	0 B/s	Idle Low Medium High Max
Device Memory			
Reads	19615307	77.885 GB/s	
Writes	30	119.118 kB/s	
Total	19615337	77.885 GB/s	Idle Low Medium High Max
ECC Overhead	10304569	40.916 GB/s	
System Memory			
[PCIe configuration: Gen2 x8, 5 Gbit/s]			
Reads	0	0 B/s	Idle Low Medium High Max
Writes	8	31.765 kB/s	Idle Low Medium High Max