Analysis Report

evolvi(configurazione, curandStateXORWOW*)

Duration	1.648 ms (1,647,949 ns)
Grid Size	[128,1,1]
Block Size	[128,1,1]
Registers/Thread	32
Shared Memory/Block	0 B
Shared Memory Requested	64 KiB
Shared Memory Executed	64 KiB
Shared Memory Bank Size	4 B

<div align="center">

[0] GeForce 840M

</div>

GPU UUID	GPU-4a39874c-8303-b7e3-9758-a265118f0297
Compute Capability	5.0
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	32
Single Precision FLOP/s	863.232 GigaFLOP/s
Double Precision FLOP/s	26.976 GigaFLOP/s
Number of Multiprocessors	3
Multiprocessor Clock Rate	1.124 GHz
Concurrent Kernel	true
Max IPC	6
Threads per Warp	32
Global Memory Bandwidth	14.4 GB/s
Global Memory Size	2 GiB
Constant Memory Size	64 KiB
L2 Cache Size	1 MiB
Memcpy Engines	1
PCIe Generation	2
PCIe Link Rate	5 Gbit/s
PCIe Link Width	4

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 "evolvi" 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 840M". 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.
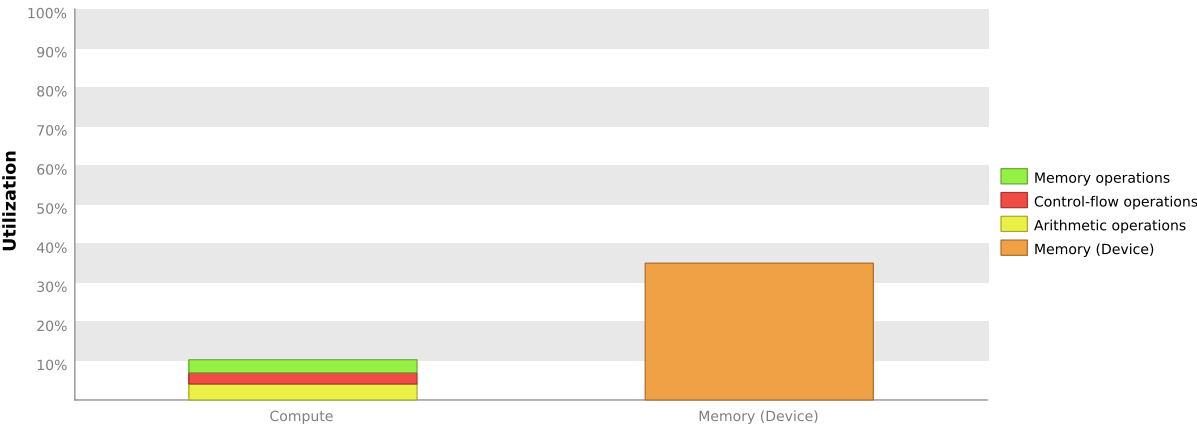
2. Instruction and Memory Latency

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 differences in the execution time of the kernel's blocks leads to poor load balancing across the SMs.
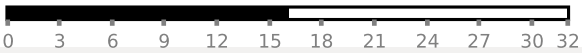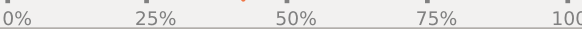
2.1. Achieved Occupancy Is Low
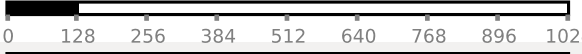
Occupancy is a measure of how many warps the kernel has active on the GPU, relative to the maximum number of warps supported by the GPU. Theoretical occupancy provides an upper bound while achieved occupancy indicates the kernel's actual occupancy. The kernel's achieved occupancy of 41.6% is significantly lower than its theoretica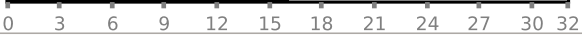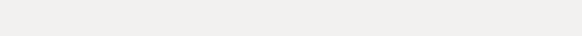l occupancy of 100%. Most likely this indicates that there is an imbalance in how the kernel's blocks are executing on the SMs so that all SMs are not equally busy over the entire execution of the kernel. The following chart shows the utilization of each multiprocessor during execution of the kernel.

Optimization: Make sure that all blocks are doing roughly the same amount of work. It may also help to increase the number of blocks executed by the kernel.
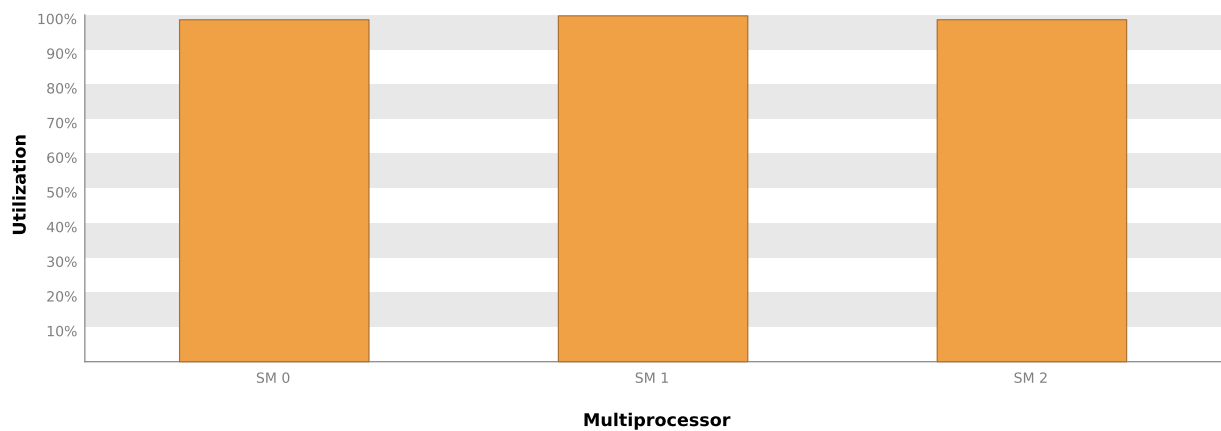

2.2. Occupancy Is Not Limiting Kernel Performance

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

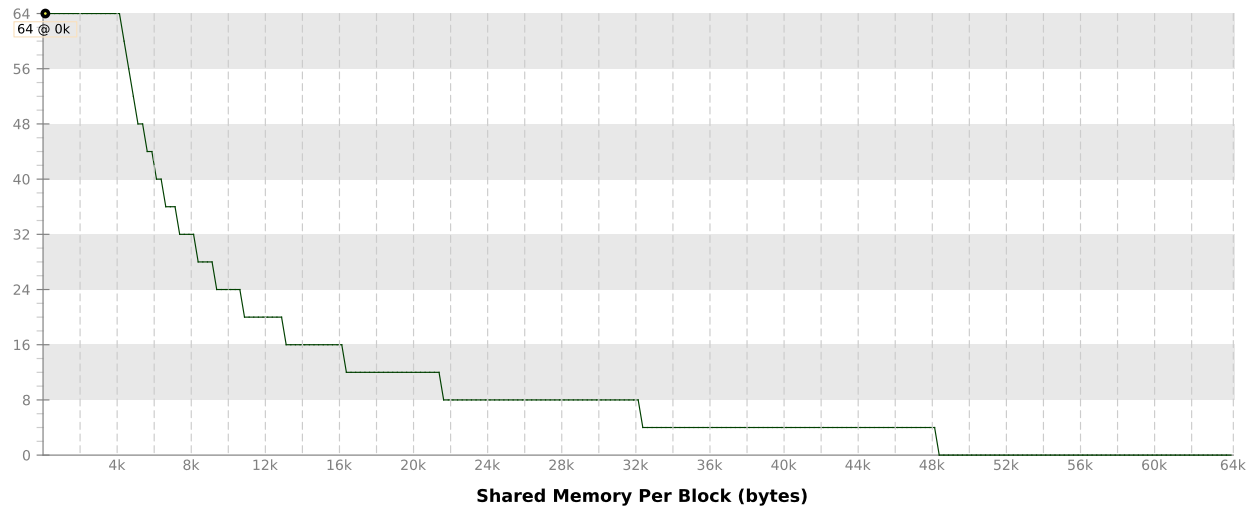| Variable	Achieved	Theoretical	Device Limit	Grid Size: [128,1,1] (128 blocks) Block Size: [128,1,1] (128 threa
Occupancy Per SM				
Active Blocks		16	32	
Active Warps	26.64	64	64	
Active Threads		2048	2048	
Occupancy	41.6%	100%	100%	
Warps				
Threads/Block		128	1024	
Warps/Block		4	32	
Block Limit		16	32	
Registers				
Registers/Thread		32	255	
Registers/Block		4096	65536	
Block Limit		16	32	
Shared Memory				
Shared Memory/Block		0	65536	
Block Limit			32	

2.3. Occupancy Charts

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

Varying Block Size

64 @ 128

(Y-axis: Warps Per SM; X-axis: Threads Per Block)
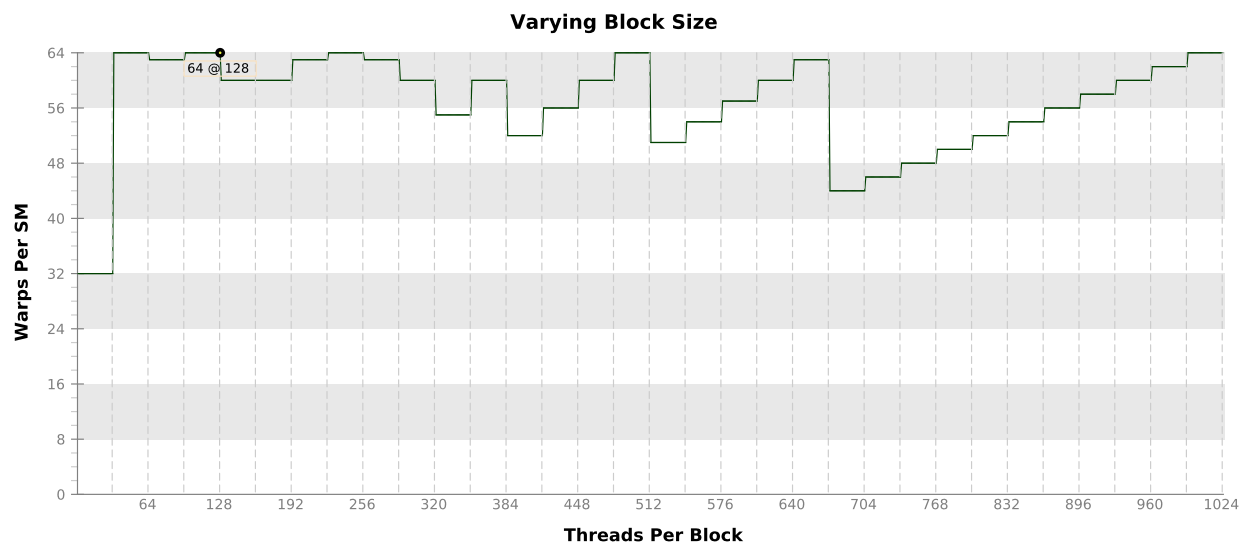
Varying Register Count

Registers Per Thread
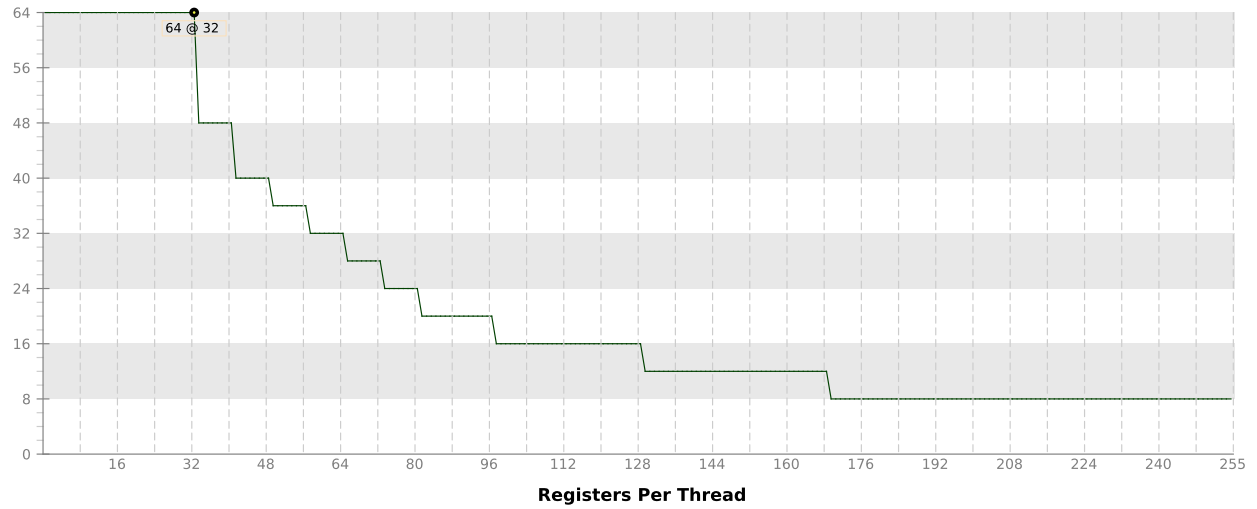
Varying Shared Memory Usage

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. Low Warp Execution Efficiency

Warp execution efficiency is the average percentage of active threads in each executed warp. Increasing warp execution efficiency will increase utilization of the GPU's compute resources. The kernel's warp execution efficiency of 45.2% is less than 100% due to divergent branches and predicated instructions. If predicated instructions are not taken into account the warp execution efficiency for these kernels is 48.5%.

Optimization: Reduce the amount of intra-warp divergence and predication in the kernel.

3.2. 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/giuseppe/Documents/myCUDA/ostacoli-2D/poligoni/improved/cuda_impr/const_memory/dynamic_parallelism/reduction.cu

| Line 108	Divergence = 25% [1106 divergent executions out of 4424 total executions]
Line 165	Divergence = 100% [32768 divergent executions out of 32768 total executions]
Line 165	Divergence = 100% [32768 divergent executions out of 32768 total executions]
Line 226	Divergence = 8.8% [45 divergent executions out of 512 total executions]
Line 228	Divergence = 7.2% [37 divergent executions out of 512 total executions]

3.3. 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 shared and constant memory.
 Texture - Load and store instructions for local, global, and texture memory.
 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.
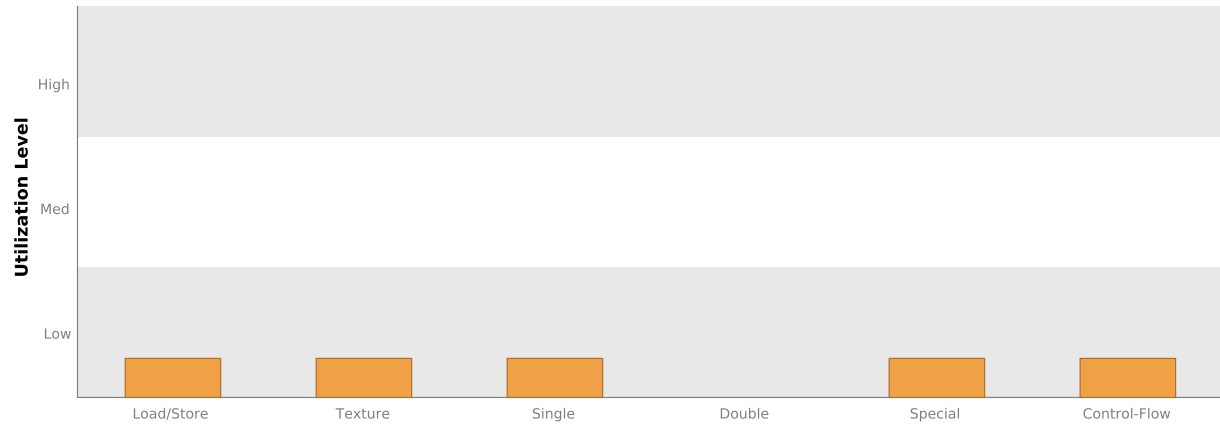
3.4. Instruction Execution Counts

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.
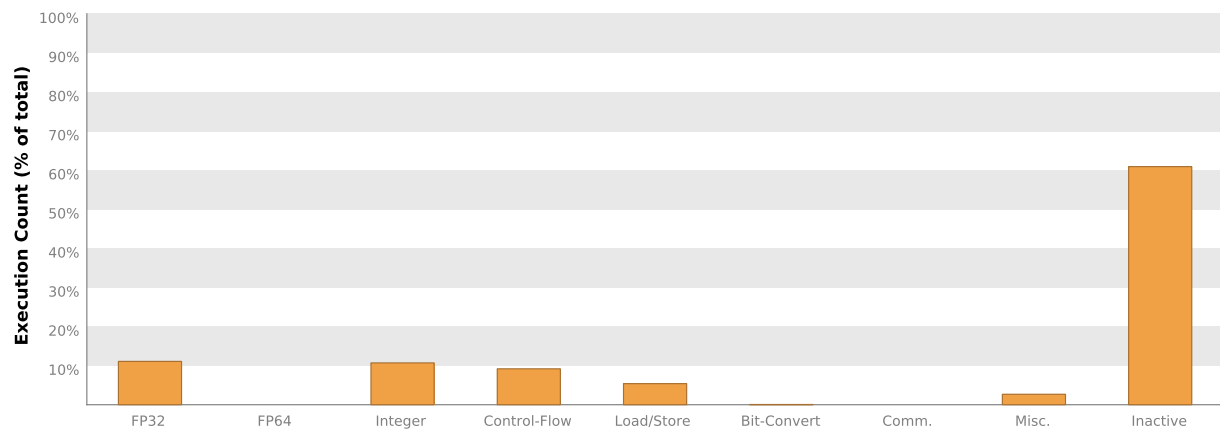
3.5. Floating-Point Operation Counts

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.
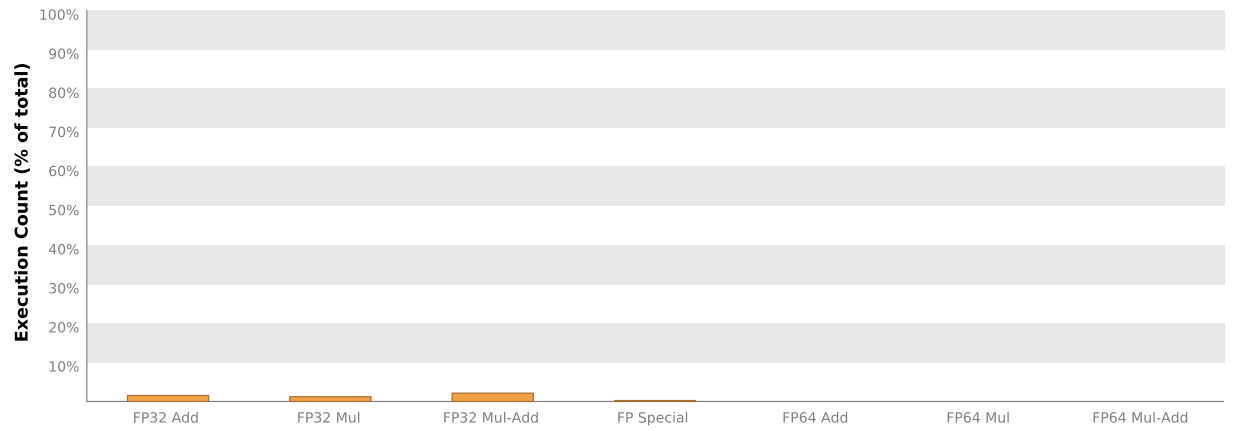
4. Memory Bandwidth

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	40675	2.936 GB/s	
Shared Stores	39770	2.871 GB/s	
Shared Total	80445	5.807 GB/s	Idle — Low — Medium — High — Max
L2 Cache			
Reads	371554	6.705 GB/s	
Writes	411403	7.424 GB/s	
Total	782957	14.129 GB/s	Idle — Low — Medium — High — Max
Unified Cache			
Local Loads	216215	3.902 GB/s	
Local Stores	130032	2.347 GB/s	
Global Loads	133828	2.283 GB/s	
Global Stores	255530	4.611 GB/s	
Texture Reads	239359	4.319 GB/s	
Unified Total	974964	17.461 GB/s	Idle — Low — Medium — High — Max
Device Memory			
Reads	116573	2.104 GB/s	
Writes	188860	3.408 GB/s	
Total	305433	5.512 GB/s	Idle — Low — Medium — High — Max
System Memory			
[PCIe configuration: Gen2 x4, 5 Gbit/s]			
Reads	0	0 B/s	Idle — Low — Medium — High — Max
Writes	4	72.182 kB/s	Idle — Low — Medium — High — Max