Open topic with navigation

You are here: Analysis Tools > CUDA Experiments > Kernel-Level Experiments > Achieved Occupancy

Achieved Occupancy

NVIDIA® Nsight™ Development Platform, Visual Studio Edition 4.6 User Guide
Send Feedback

Overview

For all CUDA kernel launches recorded in both Profile and Trace modes, the **Occupancy** experiment detail pane shows "Theoretical Occupancy", the upper limit for occupancy imposed by the kernel launch configuration and the capabilities of the CUDA device. The **Achieved Occupancy** Profile mode experiment measures occupancy during execution of the kernel, and adds the achieved values to the Occupancy experiment detail pane alongside the theoretical values. Additional graphs show achieved occupancy per SM, and illustrate how occupancy can be controlled by varying compiler and launch parameters.

Background

Definition of Occupancy

The CUDA C Programming Guide explains how a CUDA device's hardware implementation groups adjacent threads within a block into warps. A warp is considered active from the time its threads begin executing to the time when all threads in the warp have exited from the kernel. There is a maximum number of warps which can be concurrently active on a Streaming Multiprocessor (SM), as listed in the Programming Guide's table of compute capabilities. Occupancy is defined as the ratio of active warps on an SM to the maximum number of active warps supported by the SM. Occupancy varies over time as warps begin and end, and can be different for each SM.

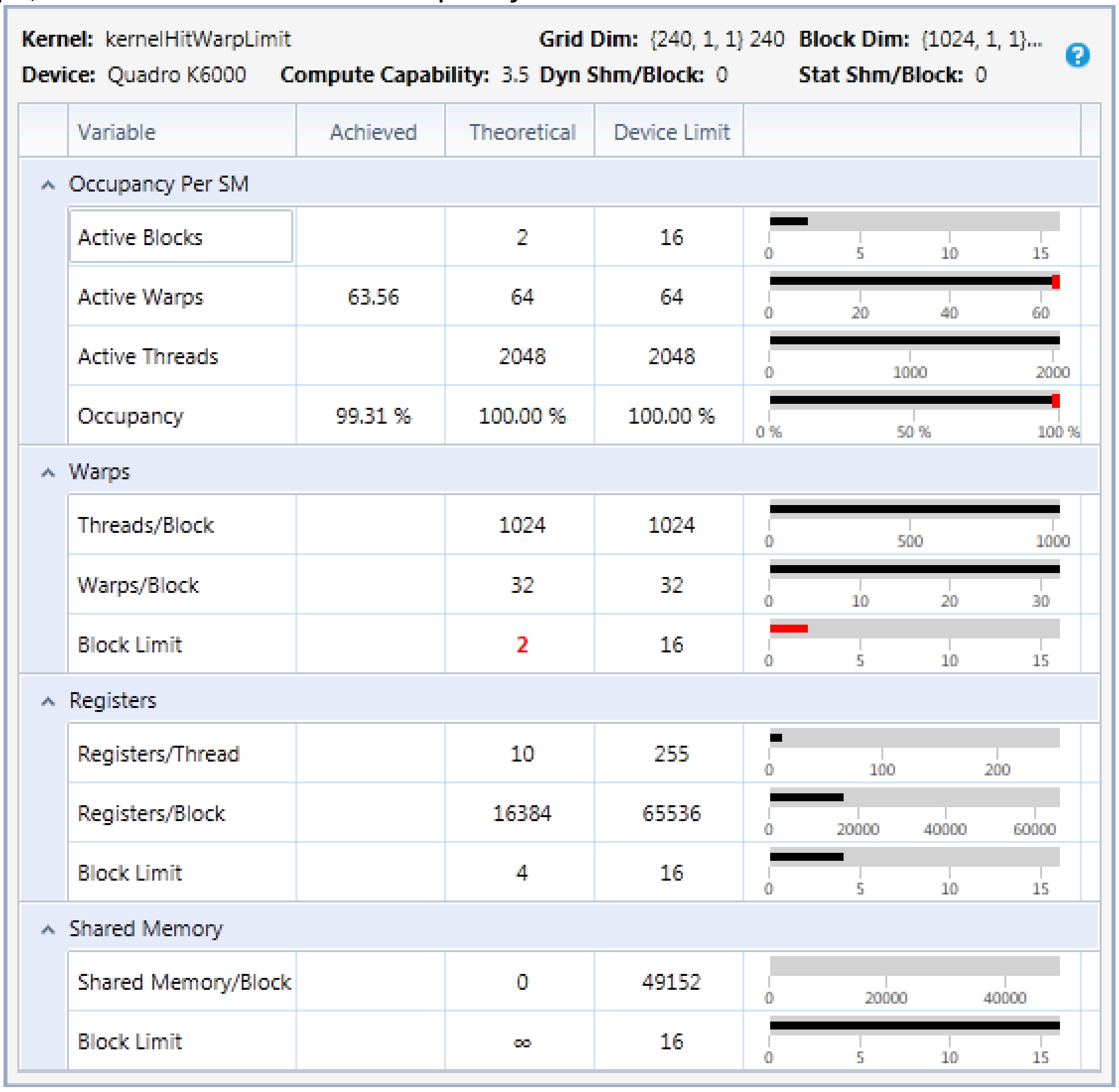
Low occupancy results in poor instruction issue efficiency, because there are not enough eligible warps to hide latency between dependent instructions. When occupancy is at a sufficient level to hide latency, increasing it further may degrade performance due to the reduction in resources per thread. An early step of kernel performance analysis should be to check occupancy and observe the effects on kernel execution time when running at different occupancy levels.

Theoretical Occupancy

There is an upper limit for active warps, and thus also for occupancy, derivable from the launch configuration, compile options for the kernel, and device capabilities. Each block of a kernel launch gets distributed to one of the SMs for execution. A block is considered active from the time its warps begin executing to the time when all warps in the block have exited from the kernel. The number of blocks which can execute concurrently on an SM is limited by the factors listed below. The upper limit for active warps is the product of the upper limit for active blocks and the number of warps per block. Thus, the upper limit for active warps can be raised by increasing the number of warps per block (defined by block dimensions), or by changing the factors limiting how many blocks can fit on an SM to allow more active blocks. The limiting factors are:

Warps per SM

The SM has a maximum number of warps that can be active at once. Since occupancy is the ratio of active warps to maximum supported active warps, occupancy is 100% if the number of active warps equals the maximum. If this factor is limiting active blocks, occupancy cannot be increased. For example, on a GPU that supports 64 active warps per SM, 8 active blocks with 256 threads per block (8 warps per block) results in 64 active warps, and 100% theoretical occupancy. Similarly, 16 active blocks with 128 threads per block (4 warps per block) would also result in 64 active warps, and 100% theoretical occupancy.

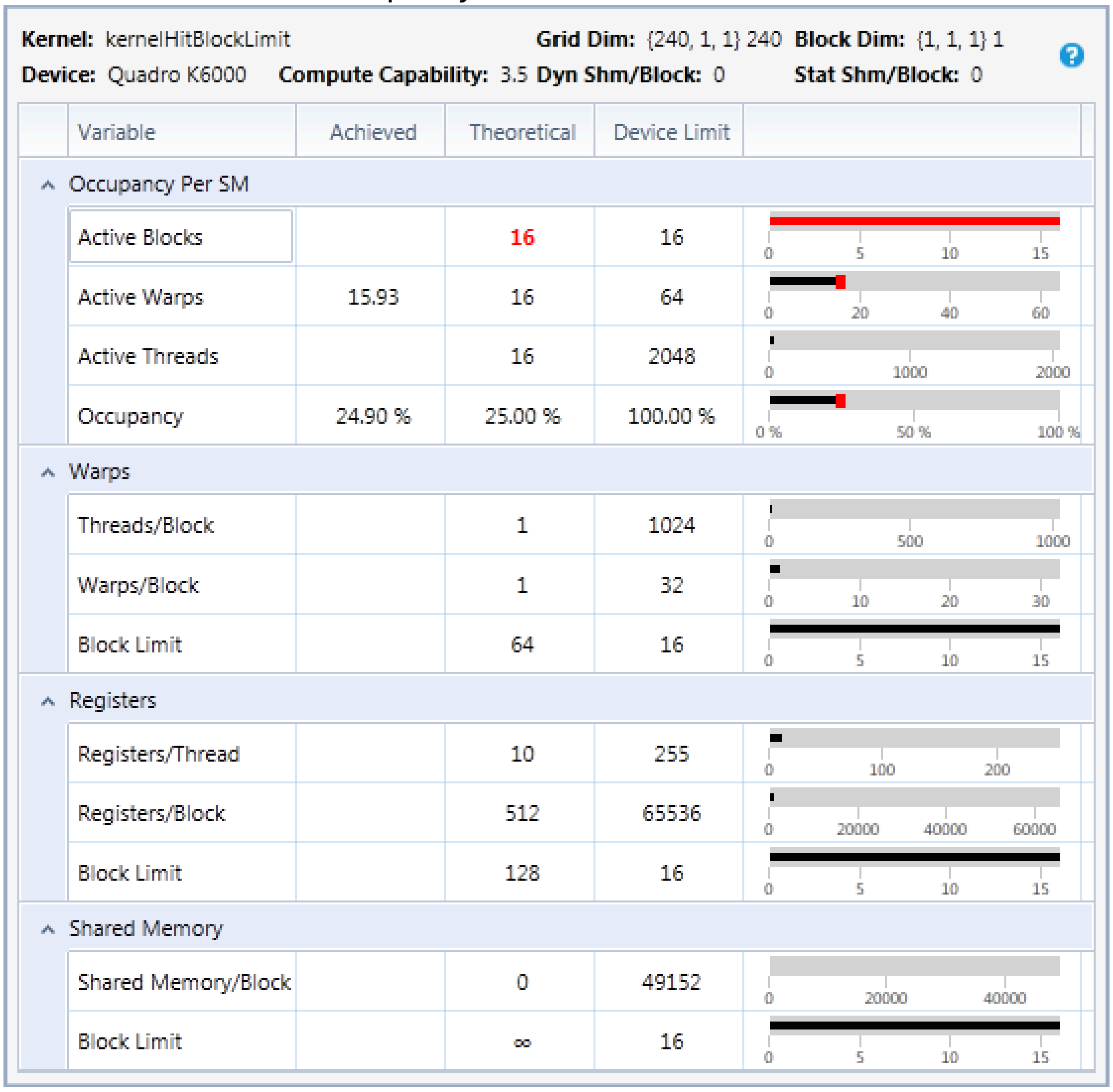


Kernel: kernelHitWarpLimit **Grid Dim:** {240, 1, 1} 240 **Block Dim:** {1024, 1, 1}...
Device: Quadro K6000 **Compute Capability:** 3.5 **Dyn Shm/Block:** 0 **Stat Shm/Block:** 0

Variable	Achieved	Theoretical	Device Limit
Occupancy Per SM			
Active Blocks		2	16
Active Warps	63.56	64	64
Active Threads		2048	2048
Occupancy	99.31 %	100.00 %	100.00 %
Warps			
Threads/Block		1024	1024
Warps/Block		32	32
Block Limit		2	16
Registers			
Registers/Thread		10	255
Registers/Block		16384	65536
Block Limit		4	16
Shared Memory			
Shared Memory/Block		0	49152
Block Limit		∞	16

Blocks per SM

The SM has a maximum number of blocks that can be active at once. If occupancy is below 100% and this factor is limiting active blocks, it means each block does not contain enough warps to reach 100% occupancy when the device's active block limit is reached. Occupancy can be increased by increasing block size. For example, on a GPU that supports 16 active blocks and 64 active warps per SM, blocks with 32 threads (1 warp per block) result in at most 16 active warps (25% theoretical occupancy), because only 16 blocks can be active, and each block has only one warp. On this GPU, increasing block size to 4 warps per block makes it possible to achieve 100% theoretical occupancy.

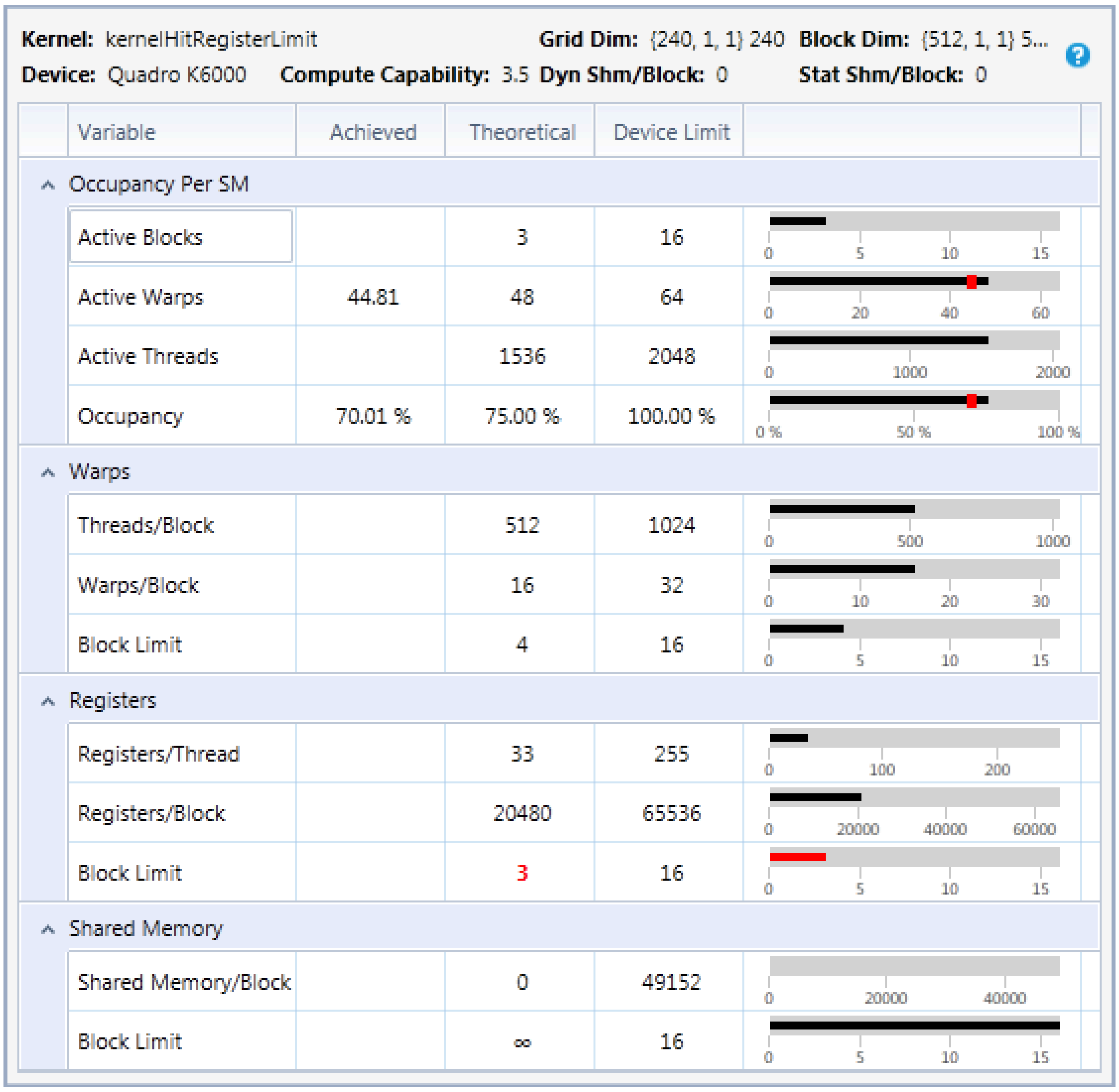


Kernel: kernelHitBlockLimit **Grid Dim:** {240, 1, 1} 240 **Block Dim:** {1, 1, 1} 1
Device: Quadro K6000 **Compute Capability:** 3.5 **Dyn Shm/Block:** 0 **Stat Shm/Block:** 0

Variable	Achieved	Theoretical	Device Limit
Occupancy Per SM			
Active Blocks		16	16
Active Warps	15.93	16	64
Active Threads		16	2048
Occupancy	24.90 %	25.00 %	100.00 %
Warps			
Threads/Block		1	1024
Warps/Block		1	32
Block Limit		64	16
Registers			
Registers/Thread		10	255
Registers/Block		512	65536
Block Limit		128	16
Shared Memory			
Shared Memory/Block		0	49152
Block Limit		∞	16

Registers per SM

The SM has a set of registers shared by all active threads. If this factor is limiting active blocks, it means the number of registers per thread allocated by the compiler can be reduced to increase occupancy (see __launch_bounds__). Kernel execution time and average eligible warps should be monitored carefully when adjusting registers per thread to control occupancy. The performance gain from improved latency hiding due to increased occupancy may be outweighed by the performance loss of having fewer registers per thread, and spilling to local memory more often. The best-performing balance of occupancy and registers per thread can be found experimentally by tracing the kernel compiled with different numbers of registers per thread, controlled via __launch_bounds__.

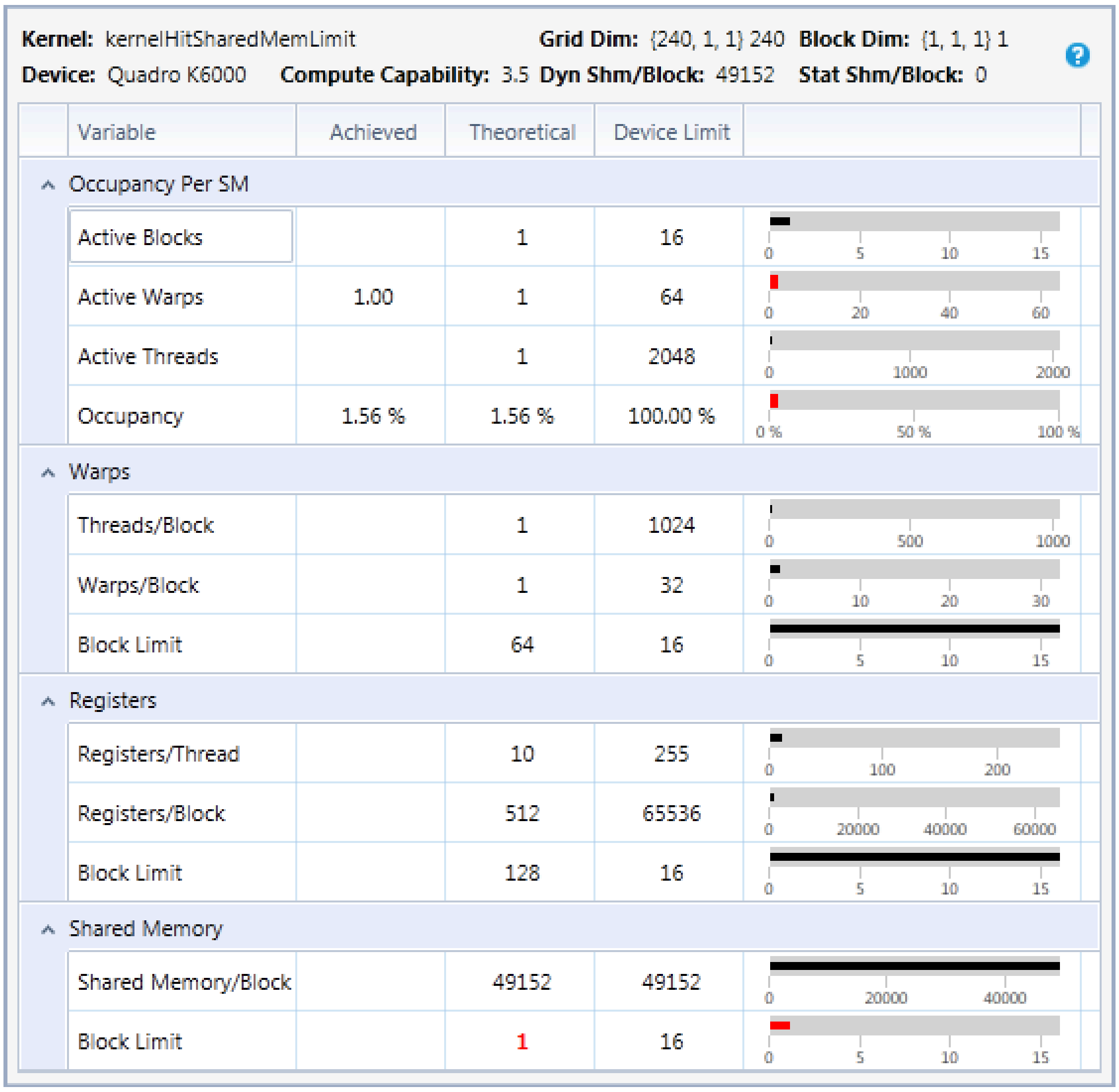


Kernel: kernelHitRegisterLimit **Grid Dim:** {240, 1, 1} 240 **Block Dim:** {512, 1, 1} 5...
Device: Quadro K6000 **Compute Capability:** 3.5 **Dyn Shm/Block:** 0 **Stat Shm/Block:** 0

Variable	Achieved	Theoretical	Device Limit
Occupancy Per SM			
Active Blocks		3	16
Active Warps	44.81	48	64
Active Threads		1536	2048
Occupancy	70.01 %	75.00 %	100.00 %
Warps			
Threads/Block		512	1024
Warps/Block		16	32
Block Limit		4	16
Registers			
Registers/Thread		33	255
Registers/Block		20480	65536
Block Limit		3	16
Shared Memory			
Shared Memory/Block		0	49152
Block Limit		∞	16

Shared Memory per SM

The SM has a fixed amount of shared memory shared by all active threads. If this factor is limiting active blocks, it means the shared memory needed per thread can be reduced to increase occupancy. Shared memory per thread is the sum of "static shared memory," the total size needed for all __shared__ variables, and "dynamic shared memory," the amount of shared memory specified as a parameter to the kernel launch. For some CUDA devices, the amount of shared memory per SM is configurable, trading between shared memory size and L1 cache size. If such a GPU is configured to use more L1 cache and shared memory is the limiting factor for occupancy, then occupancy can also be increased by choosing to use less L1 cache and more shared memory.



Kernel: kernelHitSharedMemLimit Grid Dim: {240, 1, 1} 240 Block Dim: {1, 1, 1} 1
Device: Quadro K6000 Compute Capability: 3.5 Dyn Shm/Block: 49152 Stat Shm/Block: 0

Variable	Achieved	Theoretical	Device Limit
Occupancy Per SM			
Active Blocks		1	16
Active Warps	1.00	1	64
Active Threads		1	2048
Occupancy	1.56 %	1.56 %	100.00 %
Warps			
Threads/Block		1	1024
Warps/Block		1	32
Block Limit		64	16
Registers			
Registers/Thread		10	255
Registers/Block		512	65536
Block Limit		128	16
Shared Memory			
Shared Memory/Block		49152	49152
Block Limit		1	16

Achieved Occupancy

Theoretical occupancy shows the upper bound active warps on an SM, but the true number of active warps varies over the duration of the kernel, as warps begin and end. As explained in Issue Efficiency, an SM contain one or more warp schedulers. Each warp scheduler attempts to issue instructions from a warp on each clock cycle. To sufficiently hide latencies between dependent instructions, each scheduler must have at least one warp eligible to issue an instruction every clock cycle. Maintaining as many active warps as possible (a high occupancy) throughout the execution of the kernel helps to avoid situations where all warps are stalled and no instructions are issued. Achieved occupancy is measured on each warp scheduler using hardware performance counters to count the number of active warps on that scheduler every clock cycle. These counts are then summed across all warp schedulers on each SM and divided by the clock cycles the SM is active to find the average active warps per SM. Dividing by the SM's maximum supported number of active warps gives the achieved occupancy per SM averaged over the duration of the kernel, which is shown in the Achieved Occupancy Chart. Averaging across all SMs gives the overall achieved occupancy, which is shown alongside theoretical occupancy in the experiment details pane.

Causes of Low Achieved Occupancy

Achieved occupancy cannot exceed theoretical occupancy, so the first step toward increasing occupancy should be to increase theoretical occupancy by adjusting the limiting factors. The next step is to check if the achieved value is close to the theoretical value. The achieved value will be lower than the theoretical value when the theoretical number of active warps is not maintained for the full time the SM is active. This occurs in the following situations:

Unbalanced workload within blocks

If warps within a block do not all execute for the same amount of time, the workload is said to be unbalanced. This means there are fewer active warps at the end of the kernel, which is a problem known as "tail effect". The best solution is to try having a more balanced workload among the warps in each block.

Unbalanced workload across blocks

If blocks within a grid do not all execute for the same amount of time, this workload is also unbalanced, but the efficiency of the device can be improved without having to change to a more balanced workload. Launching more blocks will allow new blocks to begin as others finish, meaning the tail effect does not occur inside every block, but only at the end of the kernel. If there are not more blocks to launch, running concurrent kernels with similar block properties can achieve the same effect.

Too few blocks launched

The upper limit for active blocks per SM is determined by the theoretical occupancy, but that calculation does not account for a launch with fewer than that number of blocks per SM. The number of SMs on the device times the maximum active blocks per SM is called a "full wave", and launching less than a full wave results in low achieved occupancy. For example, on a device with 15 SMs, and a configuration expecting 100% theoretical occupancy with 4 blocks per SM, a full wave would be 60 blocks. Launching only 45 blocks (assuming a balanced workload) will result in approximately 75% achieved occupancy.

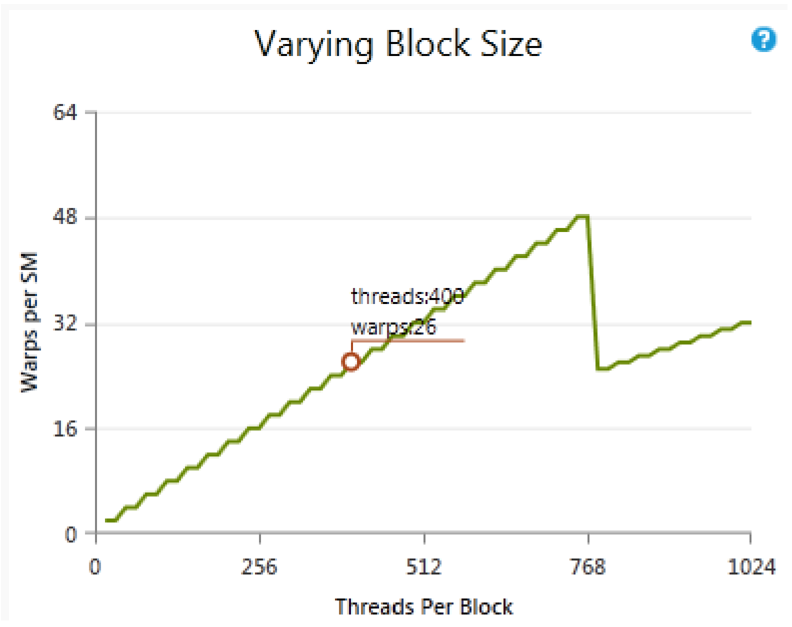
Partial last wave

The SM has a maximum number of warps that can be active at once. Since occupancy is the ratio of active warps to maximum supported active warps, occupancy is 100% if the number of active warps equals the maximum. If this factor is limiting active blocks, occupancy cannot be increased. For example, on a GPU that supports 64 active warps per SM, 8 active blocks with 256 threads per block (8 warps per block) results in 64 active warps, and 100% theoretical occupancy. Similarly, 16 active blocks with 128 threads per block (4 warps per block) would also result in 64 active warps, and 100% theoretical occupancy.

Charts

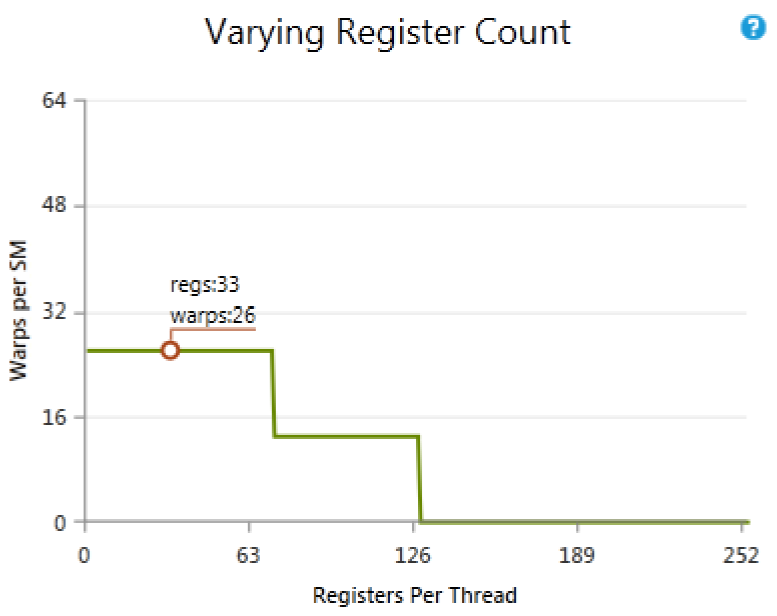
Varying Block Size

Shows how varying the block size while holding other parameters constant would affect the theoretical occupancy. The circled point shows the current number of threads per block and the current upper limit of active warps. Note that the number of active warps is not the number of warps per block (that is threads per block divided by warp size, rounded up). If the chart's line goes higher than the circle, changing the block size could increase occupancy without changing the other factors.



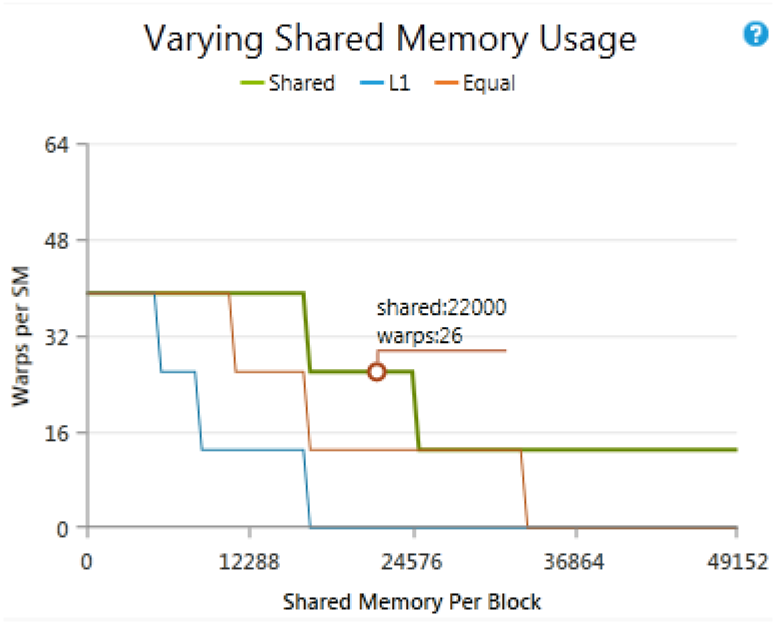
Varying Register Count

Shows how varying the register count while holding other parameters constant would affect the theoretical occupancy. The circled point shows the current number of registers per thread and the current upper limit of active warps. If the chart's line goes higher than the circle, changing the number of registers per thread could increase occupancy without changing the other factors.



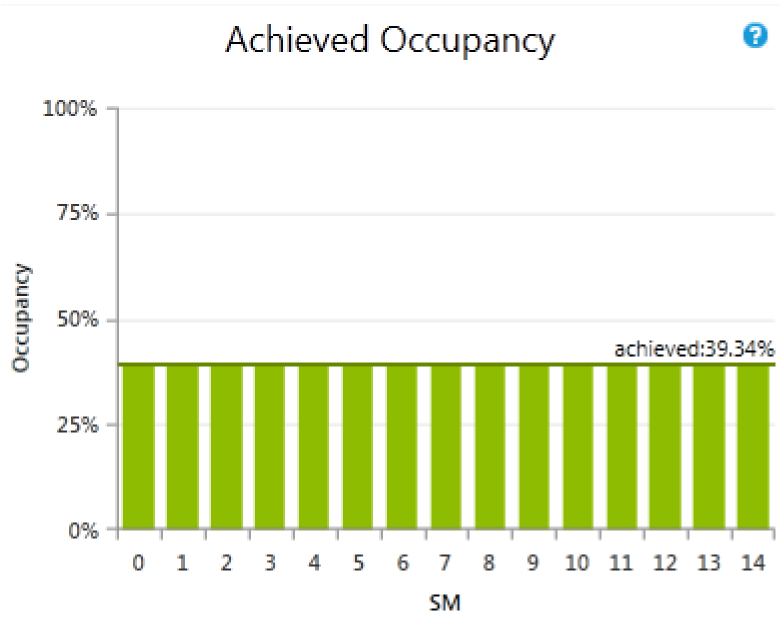
Varying Shared Memory Usage

Shows how varying the shared memory usage while holding other parameters constant would affect the theoretical occupancy. The circled point shows the current amount of shared memory per block and the current upper limit of active warps. If the chart's line goes higher than the circle, changing the amount of shared memory per block could increase occupancy without changing the other factors.



Achieved Occupancy Per SM

The achieved occupancy for each SM. The values reported are the average across all warp schedulers for the duration of the kernel execution. The line across all bars is the average, which is the number reported as Achieved Occupancy in the other tables.



Analysis

Low occupancy is not a problem in itself, but it usually results in having too few eligible warps. If the percentage of cycles with no eligible warp in the Warp Issue Efficiency chart is high …

- … try to increase the number of active warps if possible. In many cases increasing the number of active warps will result in an larger pool of eligible warps. If …
  - … the theoretical occupancy is low, try to optimize the execution configuration of the kernel launch, using the Occupancy table to identify which factor(s) are limiting occupancy. If you are register limited do not rule out experimenting with launch bounds to increase occupancy, even if this results in some register spilling.

- … the achieved occupancy is well below the theoretical occupancy, check the Instruction Statistics experiment for highly unbalanced workloads or tail effects. Potential strategies may include splitting the kernel grid in a more fine granular way, distribute work across the blocks in a more balanced way, avoiding gathering the final result on a single block, warp, or thread.
- … the Pipe Utilization experiment shows a particular pipeline is already fully utilized, increasing active warps is unlikely to results in more eligible warps, because all additional active warps will stall trying to access the oversubscribed pipeline. In this case, try to reduce the load on this pipeline or investigate if the expected peak performance for the target hardware is already reached.

Open topic with navigation