



GPU Teaching Kit
Accelerated Computing



西南石油大学 计算机科学学院
SCHOOL OF COMPUTER SCIENCE, SOUTHWEST PETROLEUM UNIVERSITY



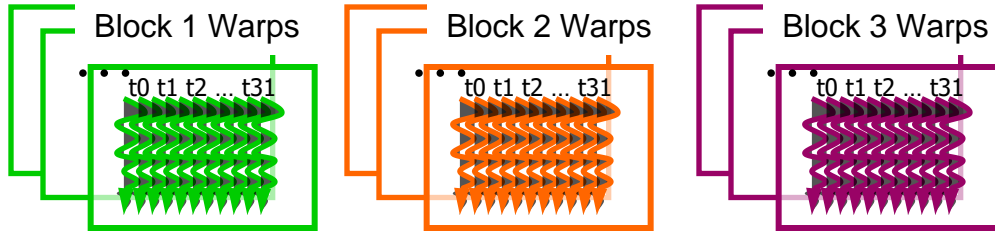
Module 5.1 – Thread Execution Efficiency

Warps and SIMD Hardware

Objective

- To understand how CUDA threads execute on SIMD Hardware
 - Warp partitioning(划分)
 - SIMD Hardware
 - Control divergence(分歧、分支)

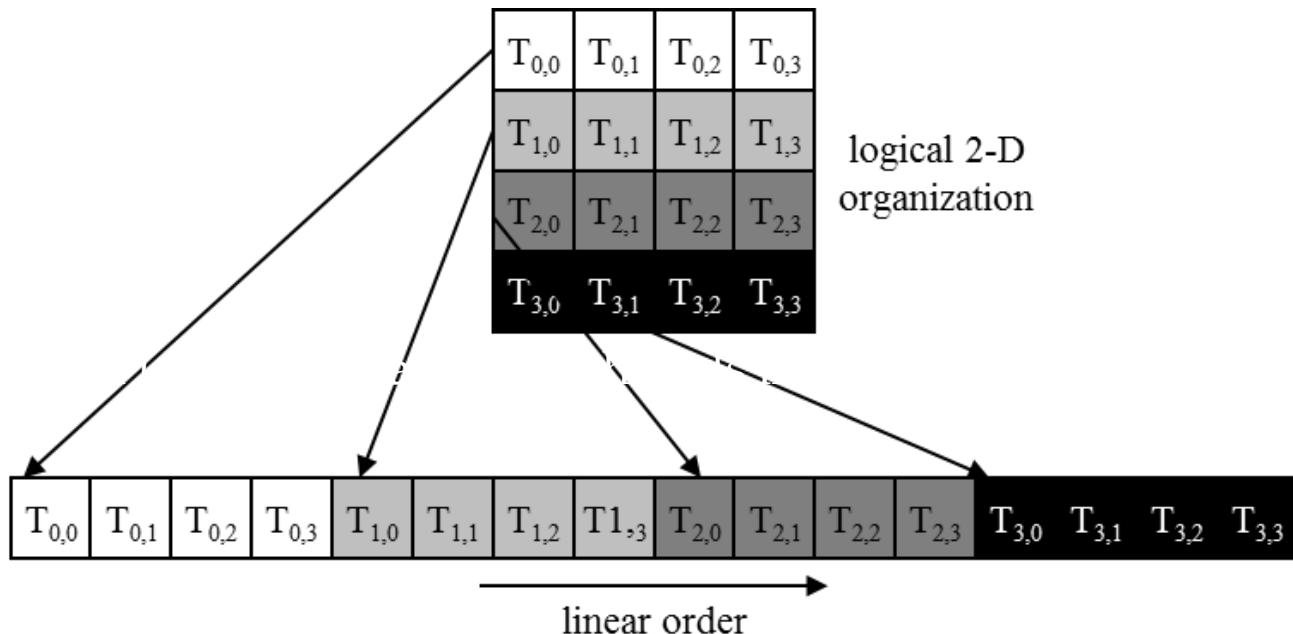
Warps as Scheduling Units



- Each block is divided into **32**-thread warps
 - An implementation technique, not part of the CUDA programming model
 - Warps are scheduling units in SM
 - Threads in a warp execute in Single Instruction Multiple Data (SIMD) manner
 - The number of threads in a warp may vary in future generations

Warps in Multi-dimensional Thread Blocks

- The thread blocks are first linearized(线性化) into 1D in row major order
 - In x-dimension first, y-dimension next, and z-dimension last

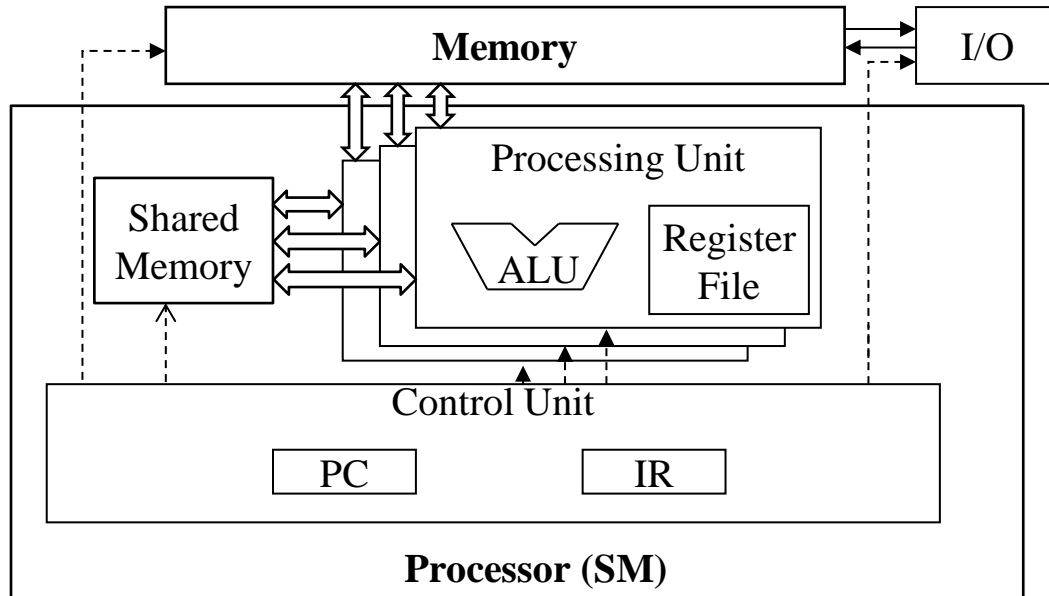


Blocks are partitioned after linearization

- Linearized thread blocks are partitioned(切分)
 - **Thread indices within a warp** are consecutive(连续的) and increasing(递增的)
 - Warp 0 starts with Thread 0
- Partitioning scheme(划分方案) is consistent(一致的) across devices
 - Thus you can use this knowledge in control flow
 - However, **the exact size of warps** may change from generation to generation
- DO NOT rely on any ordering within or between warps
 - If there are any dependencies(依赖) between threads, you must `__syncthreads()` to get correct results (more later).

SMs are SIMD Processors

- **Control unit** for instruction fetch(提取), decode(译码), and control is **shared** among multiple processing units
 - Control overhead is minimized (Module 1)



SIMD Execution Among Threads in a Warp

- **All threads in a warp** must execute the same instruction at any point in time
- This works efficiently if all threads **follow the same control flow path**
 - All **if-then-else statements** make the same decision
 - All loops iterate the same number of times

Control Divergence

- **Control divergence** occurs when threads in a warp take different control flow paths by making different control decisions
 - Some take **the then-path** and others take **the else-path** of an if-statement
 - Some threads take **different number of loop iterations** than others
- The execution of threads taking different paths are **serialized(串行化的)** in current GPUs
 - The control paths taken by the threads in a warp are traversed(遍历) one at a time until there is no more.
 - During the execution of each path, **all threads taking that path will be executed in parallel**
 - The number of different paths can be large when considering nested(嵌套的) control flow statements

Control Divergence Examples

- Divergence can arise when **branch or loop condition is a function of thread indices**
- Example kernel statement with divergence:
 - `if (threadIdx.x > 2) { }`
 - This creates two different control paths for threads in a block
 - Decision granularity(决策粒度) < **warp size**; threads **0, 1** and **2** follow different path than the rest of the threads in the first warp
- Example without divergence:
 - `If (blockIdx.x > 2) { }`
 - Decision granularity is a multiple of blocks size; all threads in any given warp follow the same path

Example: Vector Addition Kernel

Device Code

```
// Compute vector sum  $C = A + B$   
// Each thread performs one pair-wise addition  
  
__global__  
void vecAddKernel(float* A, float* B, float* C,  
    int n)  
{  
    int i = threadIdx.x + blockDim.x * blockIdx.x;  
    if(i < n) C[i] = A[i] + B[i];  
}
```

Analysis for vector size of 1,000 elements

- Assume that block size is 256 threads
 - 8 warps in each block
- **All threads** in Blocks 0, 1, and 2 are within valid range
 - i values from 0 to 767
 - There are 24 warps in these three blocks, none will have control divergence
- **Most warps** in Block 3 will not control divergence
 - Threads in the warps 0-6 are all within valid range, thus no control divergence
- One warp in Block 3 will have control divergence
 - Threads with i values 992-999 will all be within valid range
 - Threads with i values of 1000-1023 will be outside valid range
- Effect of serialization on control divergence will be small
 - 1 out of 32 warps has control divergence
 - The impact on performance will likely be less than 3%



GPU Teaching Kit
Accelerated Computing



西南石油大学 计算机科学学院
SCHOOL OF COMPUTER SCIENCE, SOUTHWEST PETROLEUM UNIVERSITY



Module 5.2 – Thread Execution Efficiency

Performance Impact of Control Divergence

Objective

- **To learn to analyze the performance impact of control divergence**
 - Boundary condition checking
 - Control divergence is data-dependent

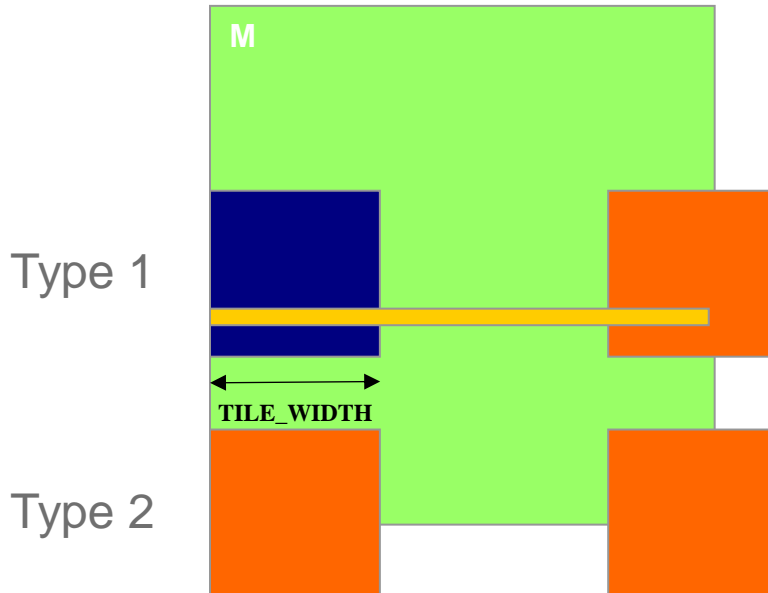
Performance Impact of Control Divergence

- Boundary condition checks are vital(至关重要) for complete functionality(完整功能) and robustness(鲁棒性) of parallel code
 - The tiled matrix multiplication kernel has many boundary condition checks
 - The concern is that these checks may cause significant performance degradation(下降)
 - For example, see the tile loading code below:

```
if(Row < Width && t * TILE_WIDTH+tx < Width) {  
    ds_M[ty][tx] = M[Row * Width + p * TILE_WIDTH + tx];  
} else {  
    ds_M[ty][tx] = 0.0;  
}  
  
if (p*TILE_WIDTH+ty < Width && Col < Width) {  
    ds_N[ty][tx] = N[(p*TILE_WIDTH + ty) * Width + Col];  
} else {  
    ds_N[ty][tx] = 0.0;  
}
```

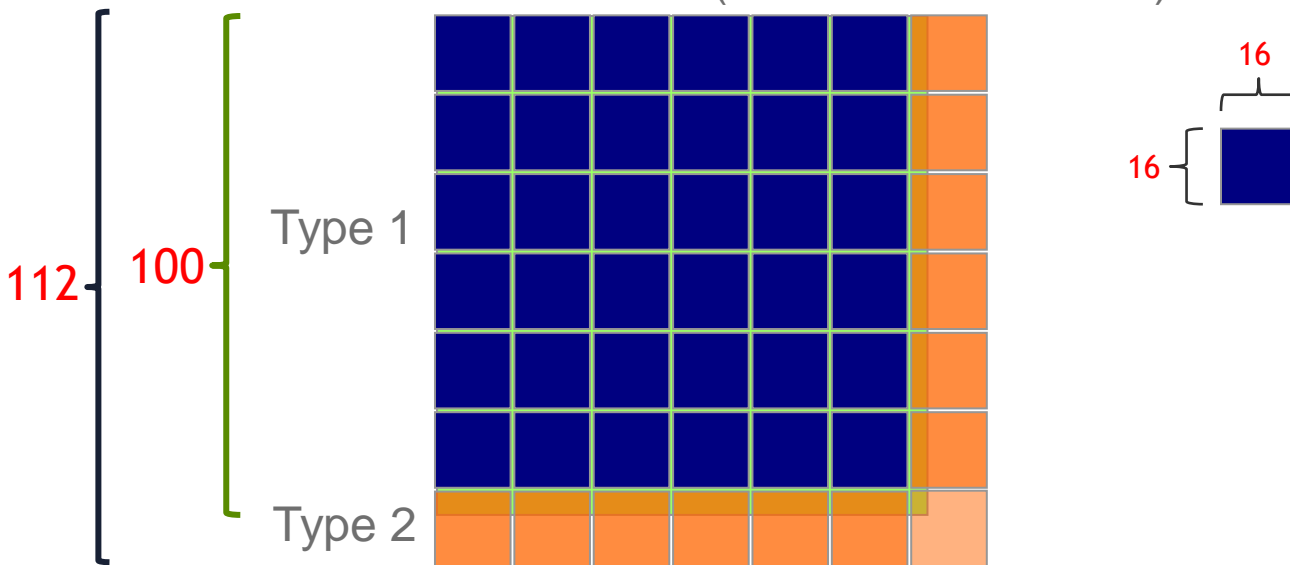
Two types of blocks in loading M Tiles

- 1. Blocks whose tiles are all within valid range until the last phase.
- 2. Blocks whose tiles are partially(部分地) outside the valid range all the way



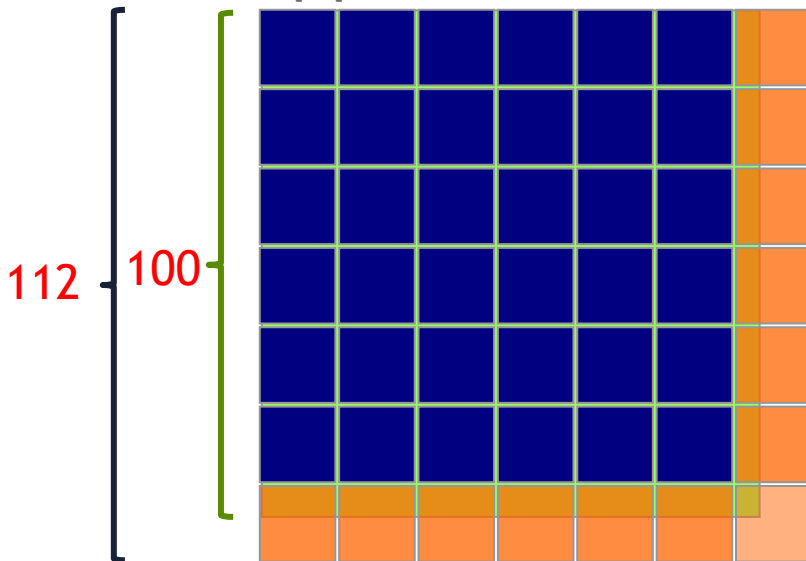
Analysis of Control Divergence Impact

- Assume **16x16** tiles and thread blocks
- Each thread block has 8 warps (**256/32**)
- Assume square matrices of **100x100**
- Each warp will go through **7 phases** (ceiling of $100/16$)
- There are **49** thread blocks (**7** in each dimension)

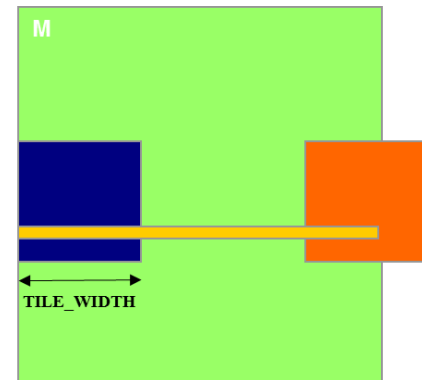


Control Divergence in Loading M Tiles

- There are **42 (6*7)** Type 1 blocks, with a total of **336 (8*42)** warps
- They all have **7 phases**, so there are **2,352 (336*7)** warp-phases
- The warps have control divergence only in their last phase
- **336 warp-phases have control divergence**

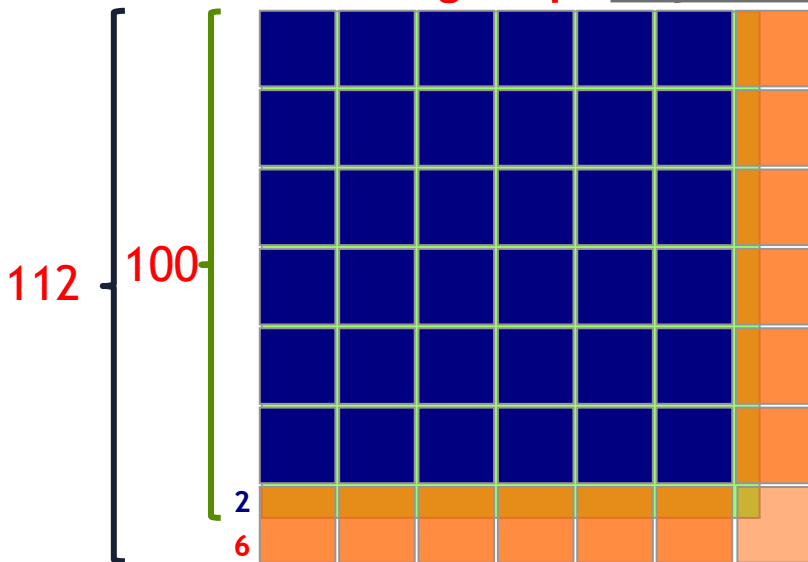


Type 1

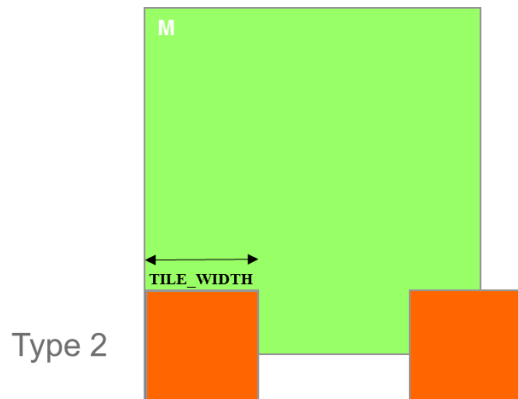


Control Divergence in Loading M Tiles (Type 2)

- Type 2: the **7 block** assigned to load the bottom tiles, with a total of **56 ($8*7$) warps**
- They all have **7 phases**, so there are **392 ($56*7$) warp-phases**
- **The first 2 warps** in each Type 2 block will stay within the valid range until the last phase
- **The 6 remaining warps** stay outside the valid range

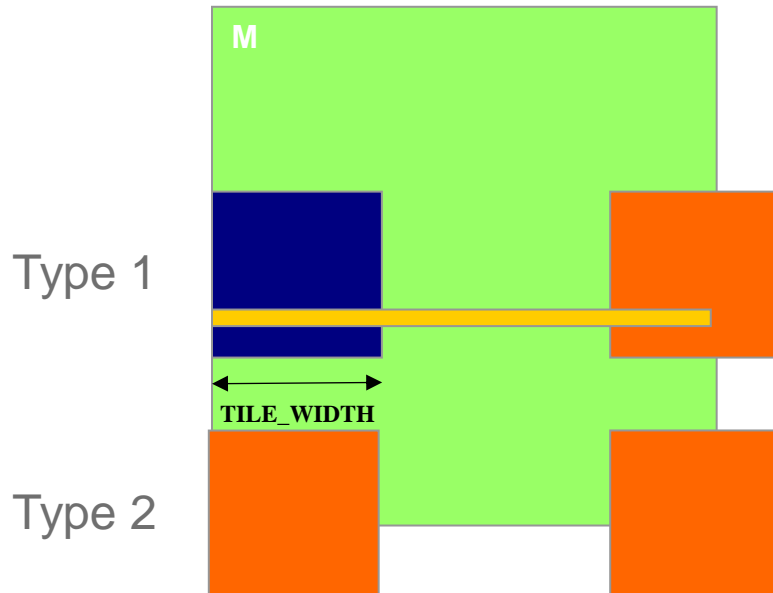


14 ($2*7$) warp-phases
have control divergence



Overall Impact of Control Divergence

- Type 1 Blocks: **336** out of **2,352** warp-phases have control divergence
- Type 2 Blocks: **14** out of **392** warp-phases have control divergence
- The performance impact is expected to be less than **12%** ($350/2,944$ or $(336+14)/(2352+392)$)



Additional Comments

- The calculation of impact of control divergence in loading N tiles is somewhat different and is left as an exercise
- The estimated performance impact is data dependent.
 - For larger matrices, the impact will be significantly smaller
- In general, the impact of control divergence for boundary condition checking for large input data sets should be insignificant
 - One should not hesitate to use boundary checks to ensure full functionality
- The fact that a kernel is full of control flow constructs does not mean that there will be heavy occurrence(频繁出现) of control divergence
- We will cover some algorithm patterns that naturally incur(引起) control divergence (such as parallel reduction(并行规约)) in the Parallel Algorithm Patterns modules



GPU Teaching Kit

Accelerated Computing



西南石油大学 计算机科学学院

SCHOOL OF COMPUTER SCIENCE, SOUTHWEST PETROLEUM UNIVERSITY



The GPU Teaching Kit is licensed by NVIDIA and the University of Illinois under the [Creative Commons Attribution-NonCommercial 4.0 International License](https://creativecommons.org/licenses/by-nc/4.0/).