OnVIDIA

你好!

GPU 教学套件

加速计算

南后油大 计算机科学学院

西南石油大学计算机科学学院

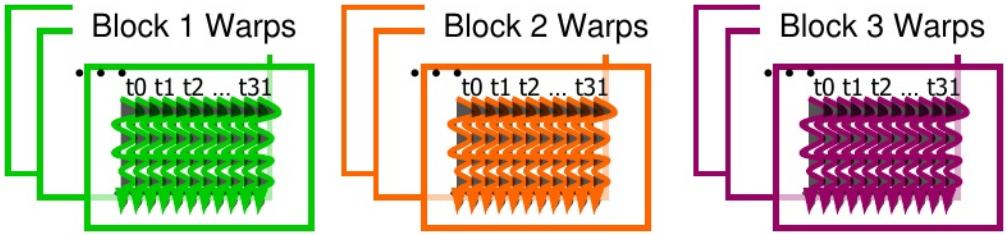
模块 5.1 - 线程执行效率

变形和单指令多数据（SIMD）硬件

目标

- 了解 CUDA 线程如何在单指令多数据（SIMD）硬件上执行
  - 沃普分区
  - 单指令多数据（SIMD）硬件
  - 控制分歧（分歧、分支）

作为调度单元的扭曲



每个线程块被划分为 32 个线程的 warp 。

一种实现技术，**并非 CUDA 编程模型的一部分**

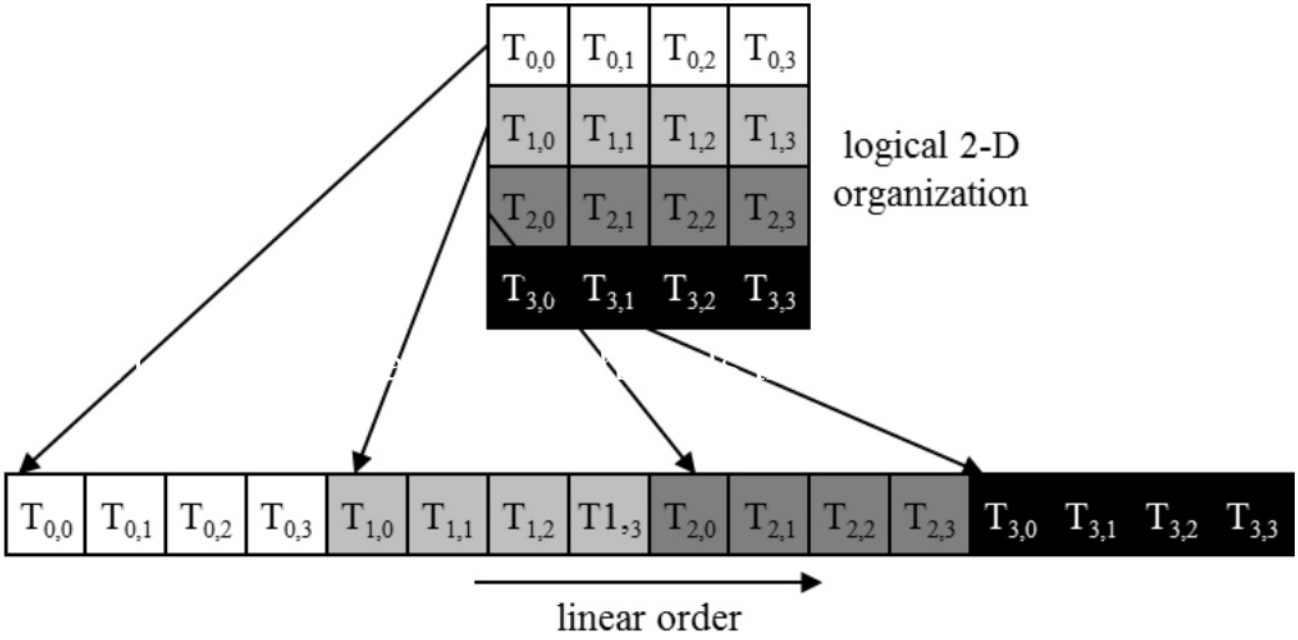
- 在 SM 中，“warp”是调度单元
- 在 warp 中的线程以**单指令多数据**（SIMD）的方式执行

在未来几代**中，一个 warp 中的线程数量**可能会有所不同。

多维线程块中的扭曲

—线程块首先按行主序线性化为 1 维

先是在 x 维度，其次是在 y 维度，最后是在 z 维度



在线性化之后对块进行划分

—线性化的线程块被切分

—在同一 warp 内的线程索引是连续且递增的

- 0 级线程从 0 号线程开始

—划分方案在各设备间是一致的

–

因此，您可以在控制流中使用这些知识。

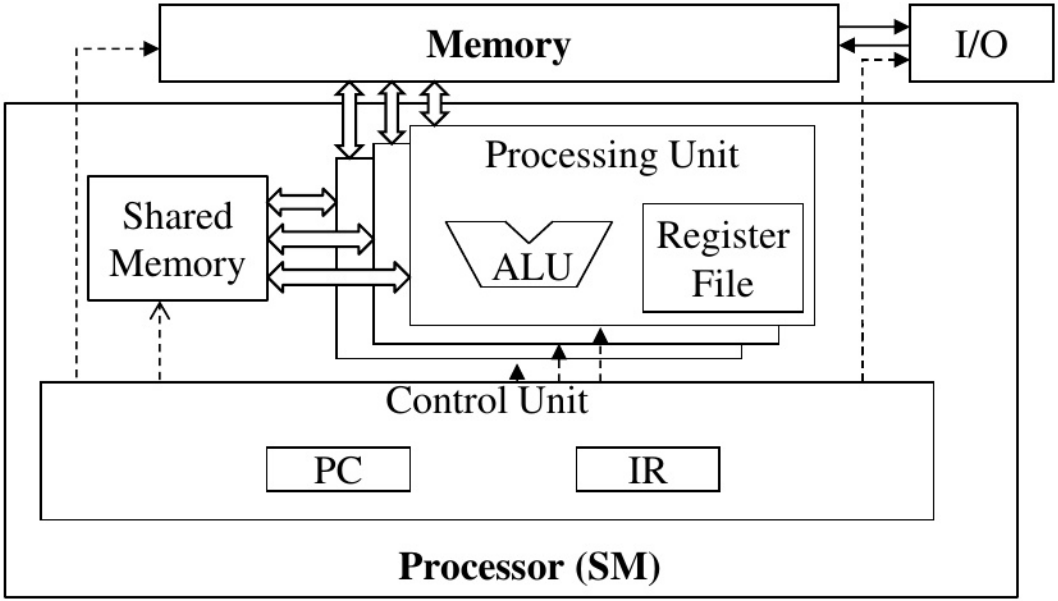
然而，翘曲的确切大小可能会在代际之间发生变化。

切勿依赖于任何在 warp 内部或 warp 之间的排序。

—如果线程之间存在任何依赖关系，您必须 __syncthreads（）以获得正确的结果（稍后会有更多内容）。

短消息处理器是单指令多数据（SIMD）处理器

- 指令提取、译码和控制的控制单元在多个处理单元之间共享
  - 控制费用被最小化了（模块 1）



线程中的单指令多数据（SIMD）执行

在同一时刻，一个 warp 中的所有线程都必须执行相同的指令。

如果所有线程都遵循相同的控制流路径，这会高效地运行。

所有的“**如果-那么-否则**”**语句**都做出相同的决定
- 所有循环的迭代次数相同

控制发散

控制**发散**发生在 warp 中的线程通过做出不同的控制决策而采取不同的控制流路径时。

- 有些人选择 if **语句中的“是”路径**，而另一些人**选择**“否”路径
- 有些线程**的循环迭代**次数**与**其他**线程不同**

—在当前的 GPU 中，走不同路径的线程的执行是串行化的。——

—在纬纱中，各股纱线所采取的控制路径会一次遍历一个，直至遍历完毕。

- 在执行每条路径期间，**所有选择该路径的线程将并行执行**。

—当考虑嵌套的控制流语句时，不同路径的数量可能会很大。

控制发散示例

当分支或循环条件是线程索引的函数时，可能会出现发散。

- 具有发散性的示例内核语句：
  - 如果（线程索引 x 大于 2）{ }

    这为块中的线程创建了两条不同的控制路径

    —决策粒度（决策粒度）< warp 大小；线程 0、1 和 2 遵循的路径与其余线程不同

    第一次变暖
- **无发散的示例：**
  - 如果（blockIdx.x > 2）{ }
    - 决策粒度是块大小的倍数；在任何给定的 warp 中，所有线程都遵循相同的路径

示例：向量相加内核

设备代码

// 计算向量之和 C = A + B

// 每个线程执行一对一的加法运算

```
__global__
void vecAddKernel(float* A, float* B, float* C,
   int n)
{
   int i = threadIdx.x + blockDim.x * blockIdx.x；
     如果（i < n），C[i] = A[i] + B[i]；
}
```

对 1000 个元素的向量大小进行分析

-假设块大小为 256 个线程

每个方块中有 8 个扭曲点

- 在 0 号、1 号和 2 号区块中的所有线程都在有效范围内。

从 0 到 767 的 -i 值

这三个街区中有 24 个扭曲点，没有一个会有控制偏差。

- 在第三区的大多数传送门将无法控制扩散。

在偏航 0 - 6 中的线程都在有效范围内，因此没有控制偏差。

- 第 3 区的一个扭曲点将出现控制偏差

i 值为 992 至 999 的线程都将处于有效范围内。

i 值为 1000 - 1023 的线程将超出有效范围。

- 序列化对控制发散的影响将很小

在 32 个华尔普中，有 1 个存在控制发散。

对性能的影响可能不到 3%。

OnVIDIA

你好!

GPU 教学套件

加速计算

南后油大 计算机科学学院

西南石油大学计算机科学学院

模块 5.2 - 线程执行效率

控制发散对性能的影响

目标

- 学会分析控制偏差对性能的影响

- 边界条件检查
- 控制发散与数据有关

控制发散对性能的影响

边界条件检查对于并行代码的完整功能和鲁棒性至关重要。

- 瓷砖矩阵乘法内核有许多边界条件的检查

令人担忧的是，这些检查可能会导致性能大幅下降。

例如，请看下面的瓷砖加载代码：

```
如果（行数 < 宽度且 t * 瓷砖宽度 + tx < 宽度）
  ds_M[ty][tx] = M[行数 * 宽度 + p * 瓷砖宽度 + tx]；
} 否则 {
 ds_M[ty][tx] = 0.0；
}
如果（p * 瓷砖宽度 + ty < 宽度且 Col < 宽度）
  ds_N[ty][tx] = N[(p * 瓷砖宽度 + ty) * 宽度 + 列数]；
} 否则 {
  ds_N[ty][tx] = 0.0；
}
```

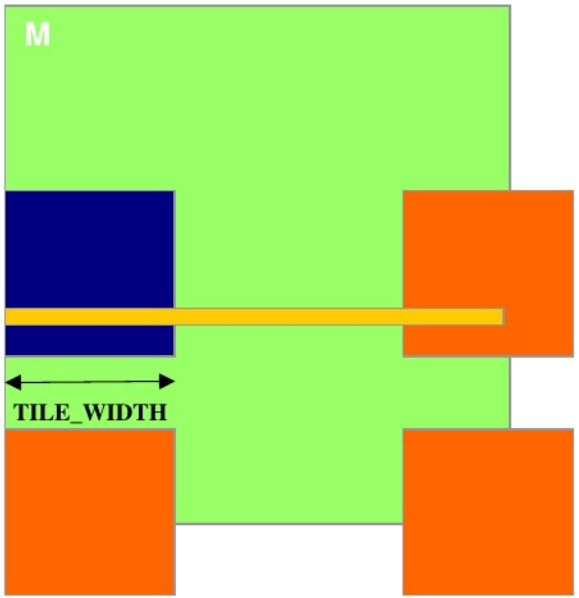
加载 M 瓷砖时的两种类型的模块

-1. 直到最后阶段其方块的瓷砖都在有效范围内的方块。

—2. 其瓷砖全部或部分始终超出有效范围的方块

1 型

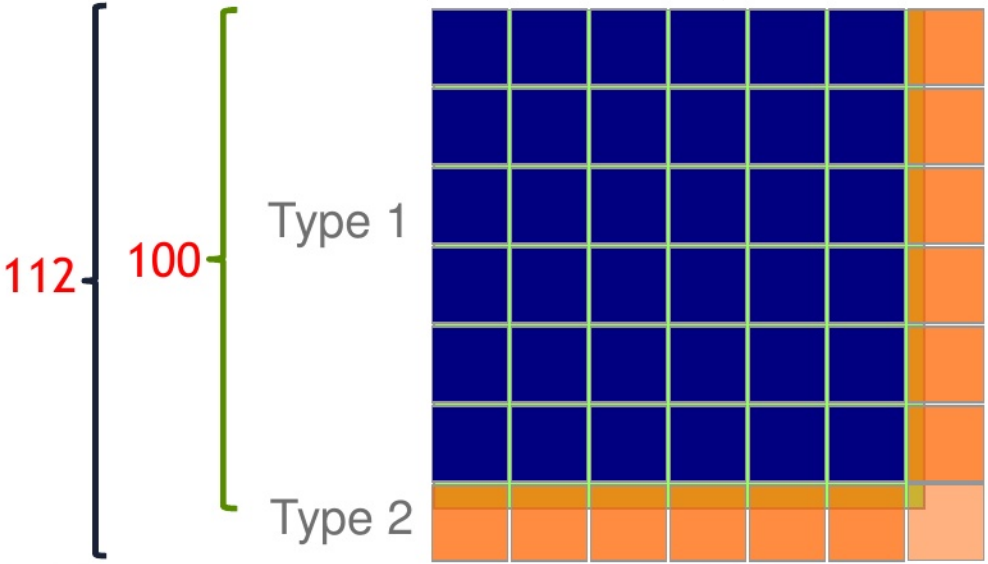
2 型

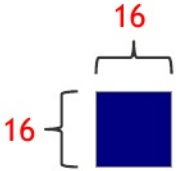


控制发散影响分析

- 假设采用 16×16 的瓷砖和线程块
- 每个线程块有 8 个 warp（256/32）
- 假设是 100×100 的方阵
- 每次变位将经历 7 个阶段（上限为 100/16）

有 49 个线程块（每个维度 7 个）





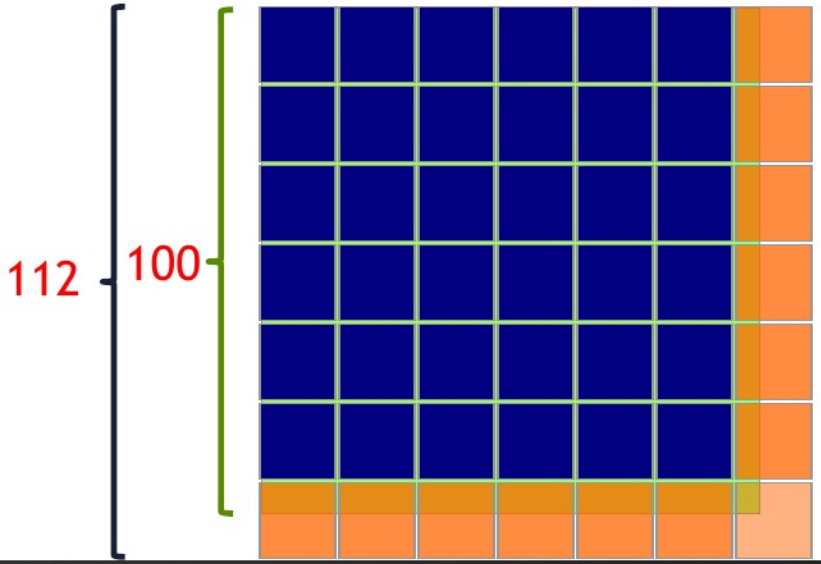
控制加载 M 级瓷砖时的偏差

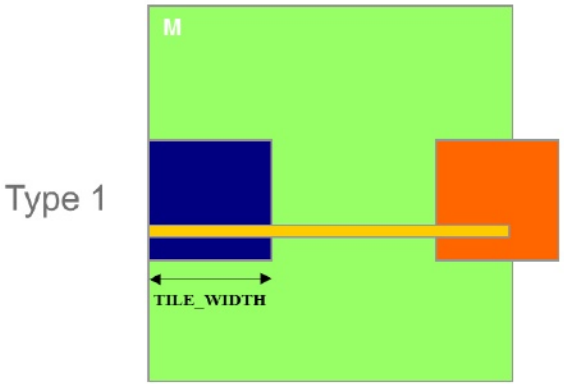
有 **42** 个（**6×7**）1 型方块，总共 **336** 个（**8×42**）传送点。

它们都有 **7** 个**阶段**，所以有 **2352** 个（**336×7**）个扭曲阶段。

这些弯道仅在其最后阶段有控制性发散。

-336 **个翘曲相位具有控制发散性**





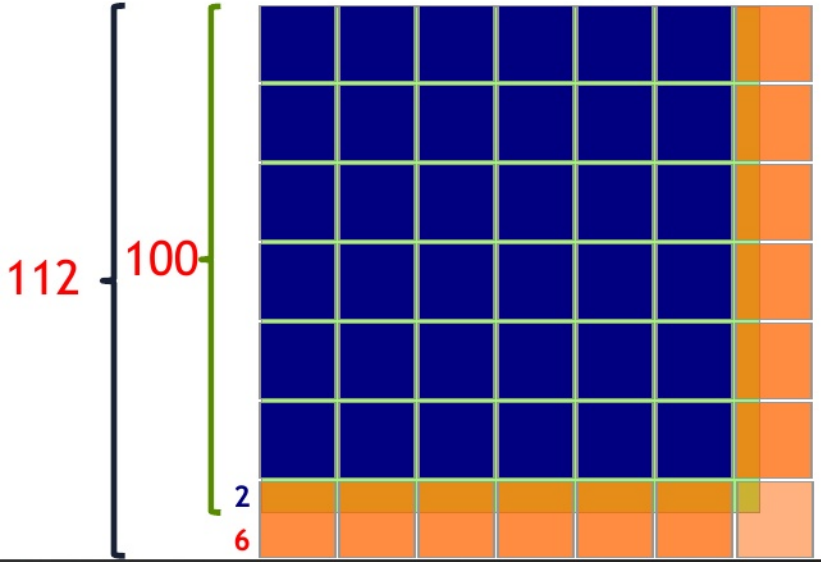
加载 M 型瓷砖时的控制偏差（类型 2）

- 类型 2：分配给加载底部瓷砖的 **7** 个**块**，总计 **56** 个**（8×7）扭曲点**

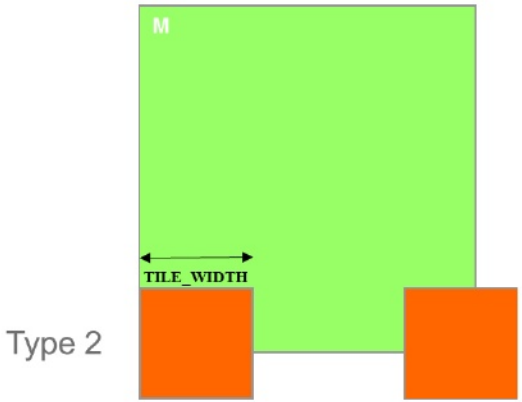
它们都有 **7** 个**阶段**，所以有 **392** 个**（56×7）扭曲阶段**。

在每个 2 型区块中，**前**两个扭曲度将保持在**有效范围内，直至最后阶段**。

剩余的 6 个翘曲度仍在**有效**范围之外。



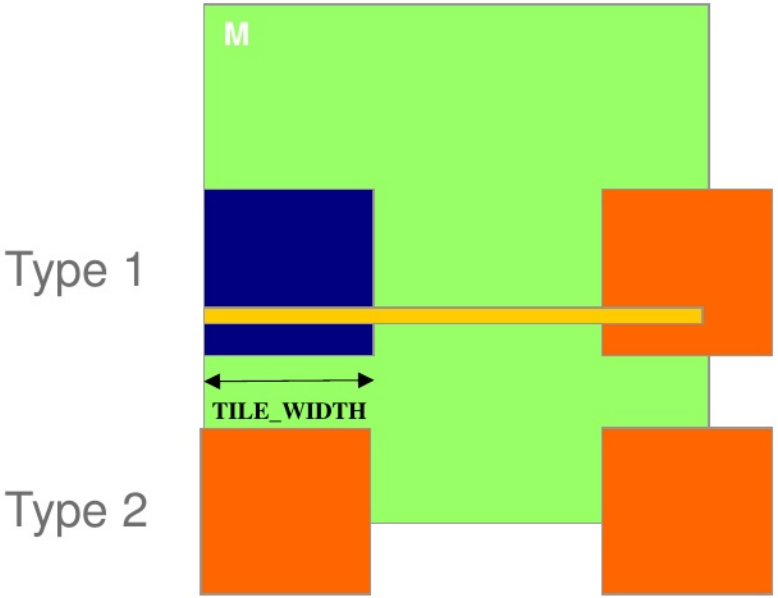
14 个（2×7）变相位存在控制偏差



控制偏差总体影响

- 1 型区块：在 **2352 个翘变相位中，有 336** 个存在控制偏差
- 2 型区块：在 **392** 个翘变相位中，有 **14** 个存在控制发散。

预计性能影响将小于 **12%（350/2944 或（336 + 14）/（2352 + 392）**



补充评论

在加载 N 个图块的情况下，控制偏差影响的计算方式有所不同，这作为一项练习留给读者去做。

预估的性能影响取决于数据。

对于较大的矩阵，其影响将会显著减小。

总的来说，对于大型输入数据集，控制发散对边界条件检查的影响应该不大。

人们不应犹豫使用边界检查来确保全部功能。

内核中充满了控制流结构这一事实并不意味着控制发散的情况会频繁出现。

—在“并行算法模式”模块中，我们将探讨一些自然会导致控制发散（例如并行规约）的算法模式。

OnVIDIA

你好!

GPU 教学套件

加速计算