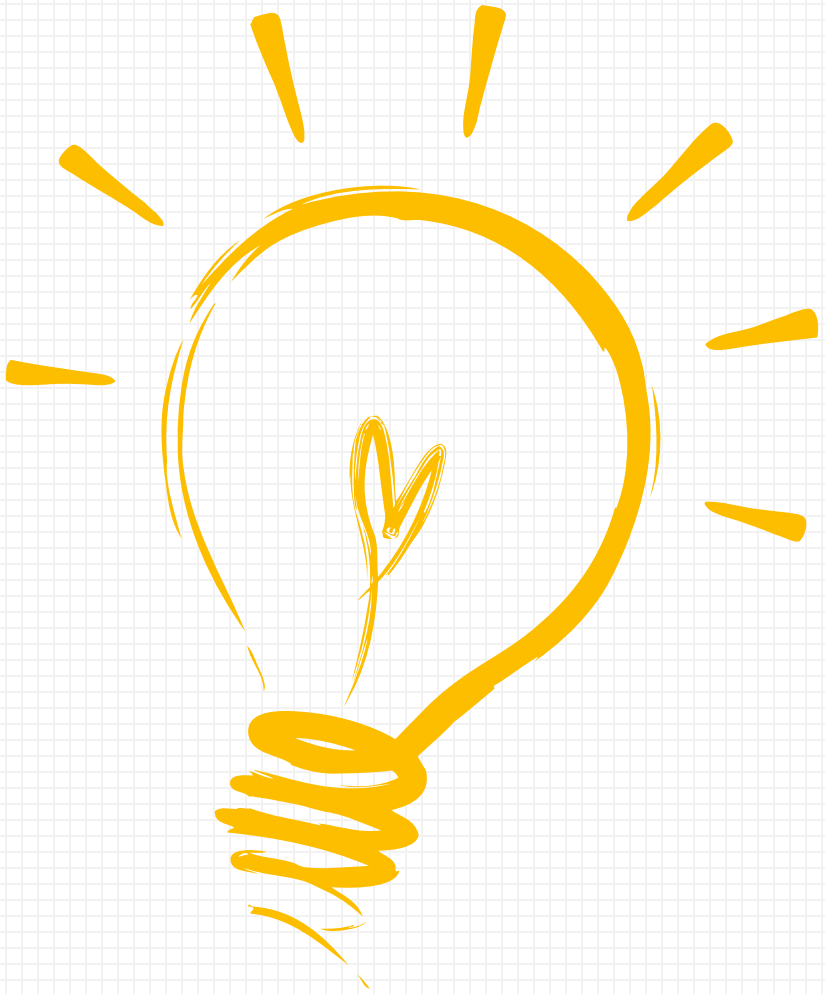
CUDA线程模型

CUDA并行编程系列课程

主讲：权双



CONTENTS

线程模型结构

★ 1、线程模型重要概念：

（1）grid　　网格

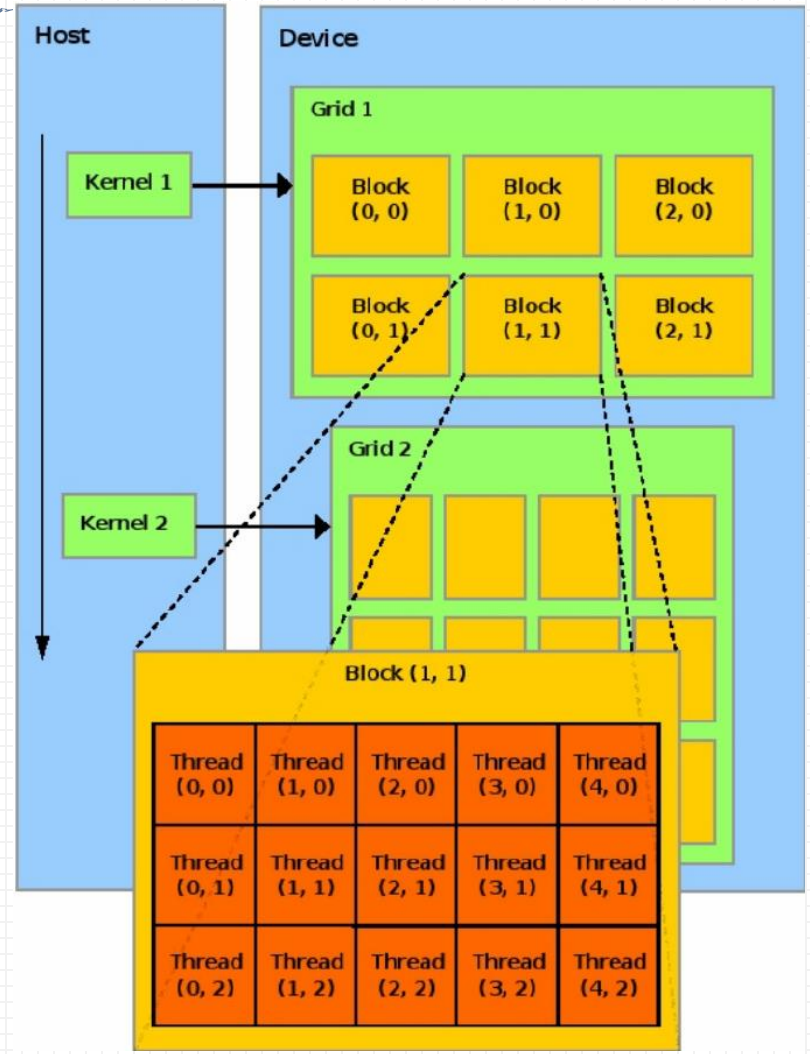
（2）block　　线程块

★ 2、线程分块是逻辑上的划分，物理上线程不分块

★ 3、配置线程：<<<grid_size，block_size>>>

★ 4、最大允许线程块大小：1024

最大允许网格大小：$2^{31}-1$ （针对一维网格）



一维线程模型

★ 1、每个线程在核函数中都有一个唯一的身份标识；

★ 2、每个线程的唯一标识由这两个<<<grid_size，block_size>>>确定；grid_size，block_size保存在内建变量（build-in variable），目前考虑的是一维的情况:

（1）gridDim.x：该变量的数值等于执行配置中变量grid_size的值；

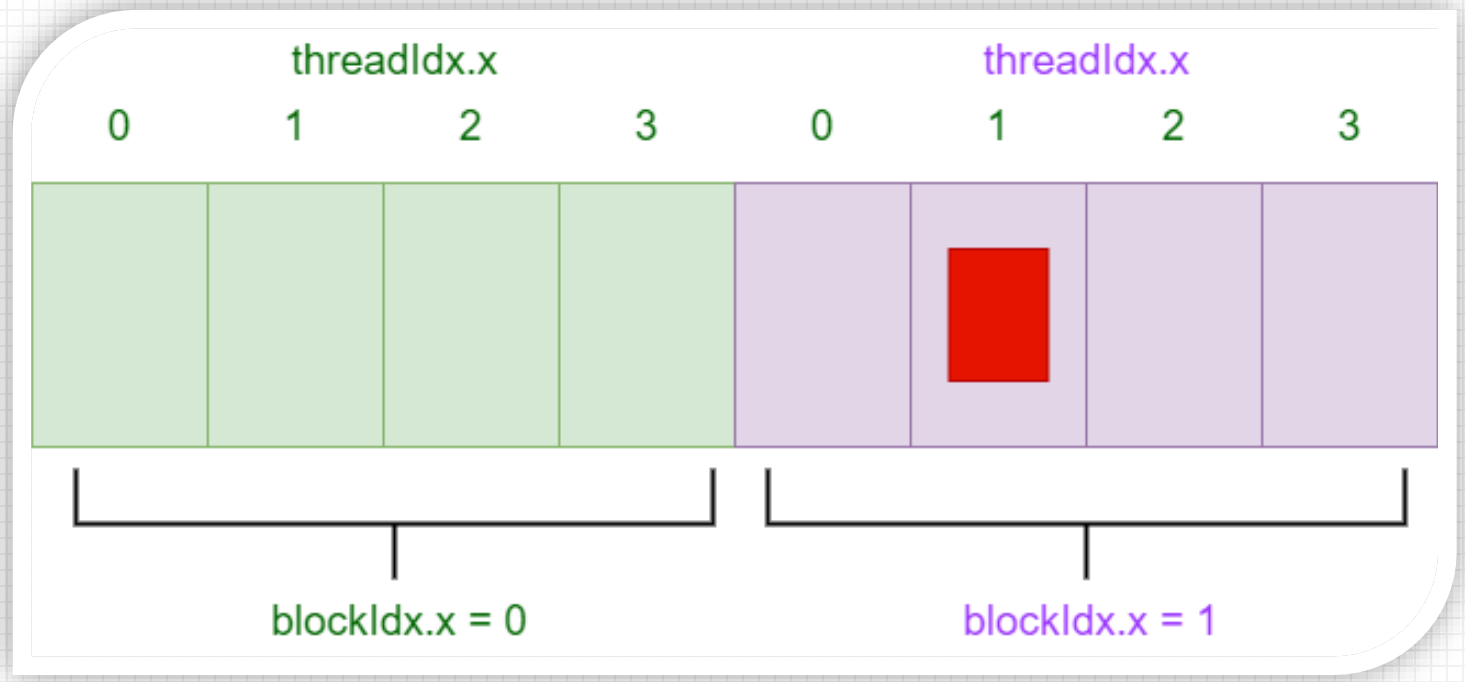
（2）blockDim.x：该变量的数值等于执行配置中变量block_size的值。

★ 3、线程索引保存成内建变量（ build-in variable）：

（1）blockIdx.x：该变量指定一个线程在一个网格中的线程块索引值，范围为0~ gridDim.x-1；

（2）threadIdx.x：该变量指定一个线程在一个线程块中的线程索引值，范围为0~ blockDim.x-1。

一维线程模型



★ 例如　　kernel_fun<<<2, 4>>>() ;

★ 线程唯一标识：

Idx = threadIdx.x + blockIdx.x * blockDim.x

★ gridDim.x 的值为2

blockDim.x的值为4

blockIdx.x取值范围为0~1

threadIdx.x取值范围为0~3

推广到多维线程

1、CUDA可以组织三维的网格和线程块；

2、blockIdx和threadIdx是类型为uint3的变量，该类型是一个结构体，具有x,y,z三个成员（3个成员都为无符号类型的成员构成）：

$$\begin{cases} blockIdx.x \\ blockIdx.y \\ blockIdx.z \end{cases} \qquad \begin{cases} \mathrm{thread}Idx.x \\ threadIdx.y \\ threadIdx.z \end{cases}$$

推广到多维线程

3、gridDim和blockDim是类型为dim3的变量，该类型是一个结构体，具有x,y,z三个成员

$$\begin{cases}\text{gridDim}.x\\ \text{gridDim}.y\\ \text{gridDim}.z\end{cases}\qquad\qquad\begin{cases}block\text{Dim}.x\\ block\text{Dim}.y\\ block\text{Dim}.z\end{cases}$$

4、取值范围

blockIdx.x 范围----------[0, gridDim.x-1]

blockIdx.y 范围----------[0, gridDim.y-1]

blockIdx.z范围----------[0, gridDim.z-1]

threadIdx.x 范围----------[0, blockDim.x-1]

threadIdx.y 范围----------[0, blockDim.y-1]

threadIdx.z 范围----------[0, blockDim.z-1]

注意：内建变量只在核函数有效，且无需定义！

推广到多维线程

<<<grid_size, block_size>>>

grid_size ----------> gridDim.x　　block_size ----------> blockDim.x

gridDim和blockDim没有指定的维度默认为1：

$$\begin{cases} \text{gridDim}.x = \text{grid\_size} \\ \text{gridDim}.y = 1 \\ \text{gridDim}.z = 1 \end{cases} \qquad \begin{cases} block\text{Dim}.x = \text{block\_size} \\ block\text{Dim}.y = 1 \\ block\text{Dim}.z = 1 \end{cases}$$

推广到多维线程

★ 定义多维网格和线程块（c++构造函数语法）：

```
dim3 grid_size(Gx, Gy, Gz);
dim3 block_size(Bx, By, Bz);
```

★ 举个例子,定义一个 2×2×1 的网格， 5×3×1的线程块，代码中定义如下:

```
dim3 grid_size(2, 2);      // 等价于dim3 grid_size(2, 2, 1);

dim3 block_size(5, 3);    // 等价于dim3 block_size(5, 3, 1);
```

推广到多维线程

★ dim3 grid_size(3, 2, 1);

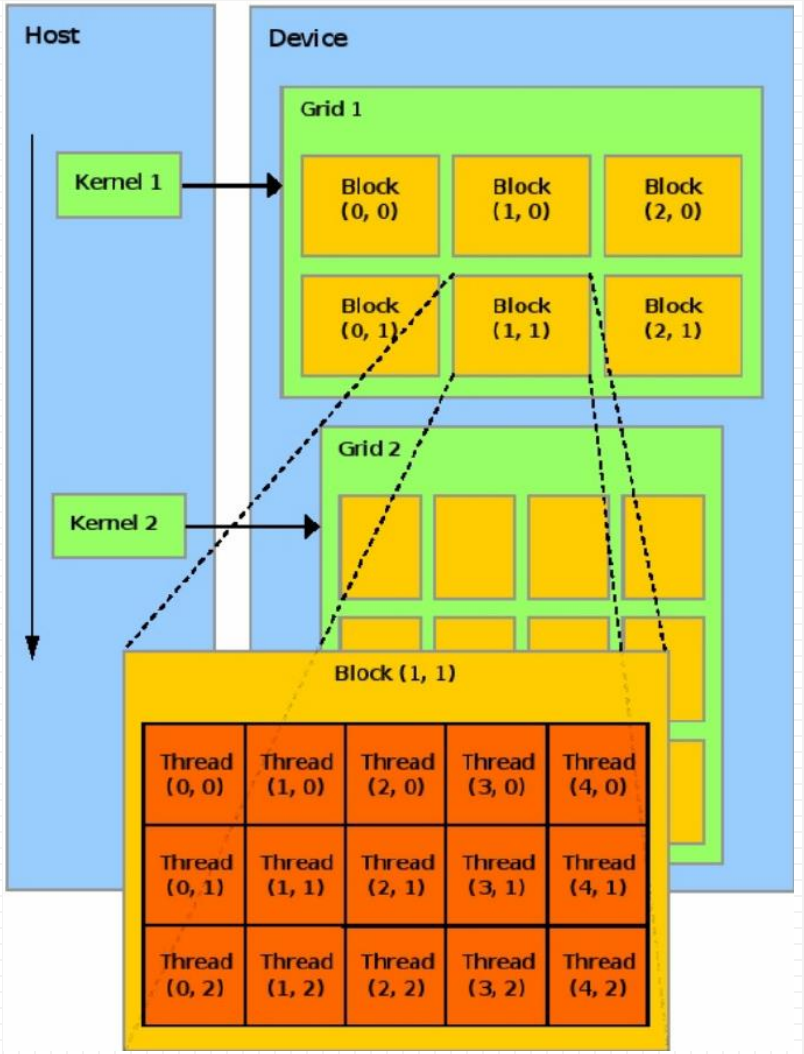
dim3 block_size(5, 3, 1);

★ 多维网格和多维线程块本质是一维的，GPU物理上不分块。

★ 每个线程都有唯一标识：

int tid = threadIdx.y * blockDim.x + threadIdx.x;

int bid = blockIdx.y * gridDim.x + blockIdx.x;



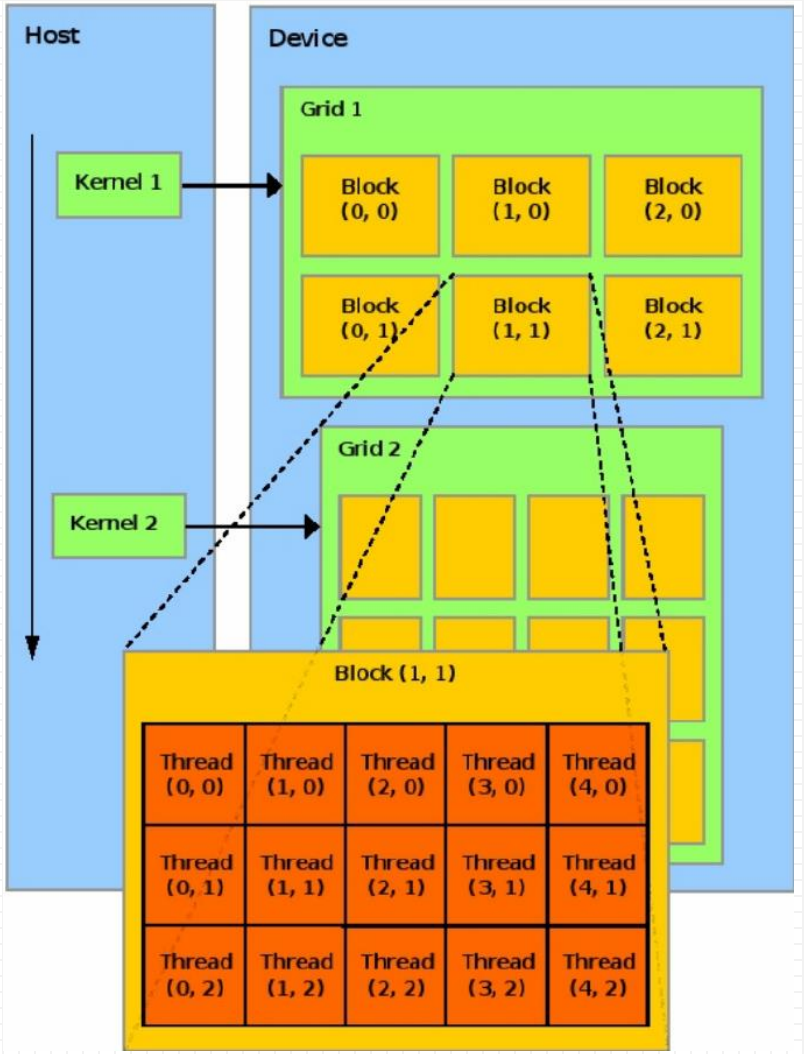
推广到多维线程

★ 多维线程块中的线程索引：

```
int tid = threadIdx.z * blockDim.x * blockDim.y +
          threadIdx.y * blockDim.x + threadIdx.x;
```

★ 多维网格中的线程块索引：

```
int bid = blockIdx.z * gridDim.x * gridDim.y +
          blockIdx.y * gridDin.x + blockIdx.x;
```



网格和线程块的限制条件

★ 网格大小限制：

gridDim.x 最大值---------- $2^{31}-1$

gridDim.y 最大值---------- $2^{16}-1$

gridDim.z 最大值---------- $2^{16}-1$

★ 线程块大小限制：

blockDim.x 最大值---------- 1024

blockDim.y 最大值---------- 1024

blockDim.z 最大值---------- 64

注意：线程块总的大小最大为1024！！

THANKS

谢谢聆听