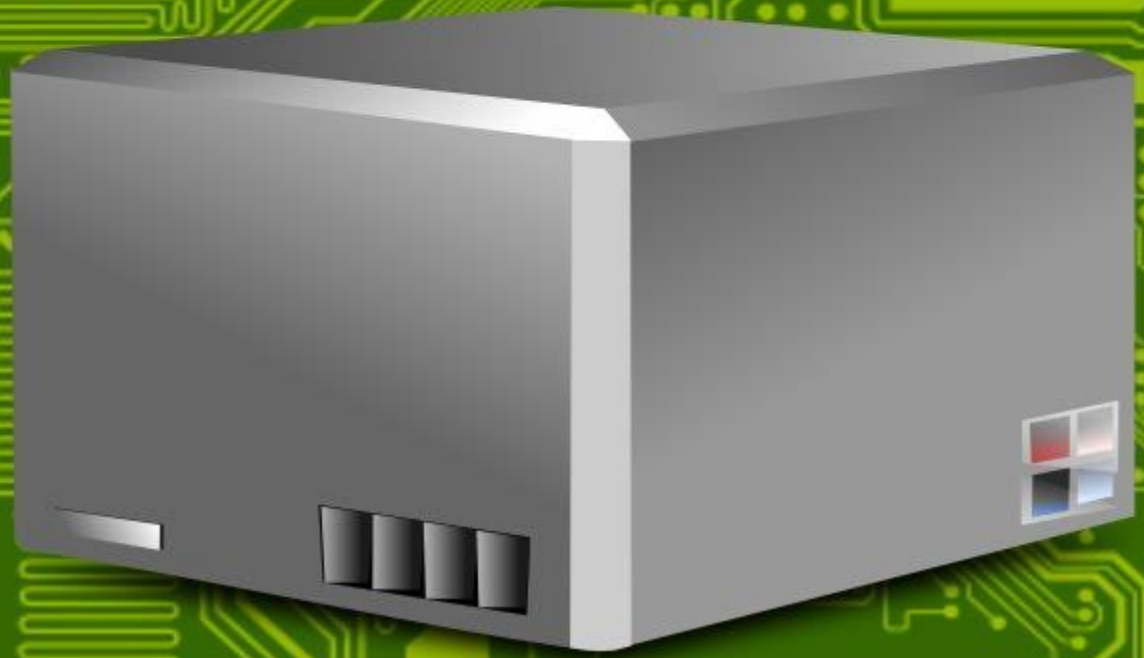
GPU Computing



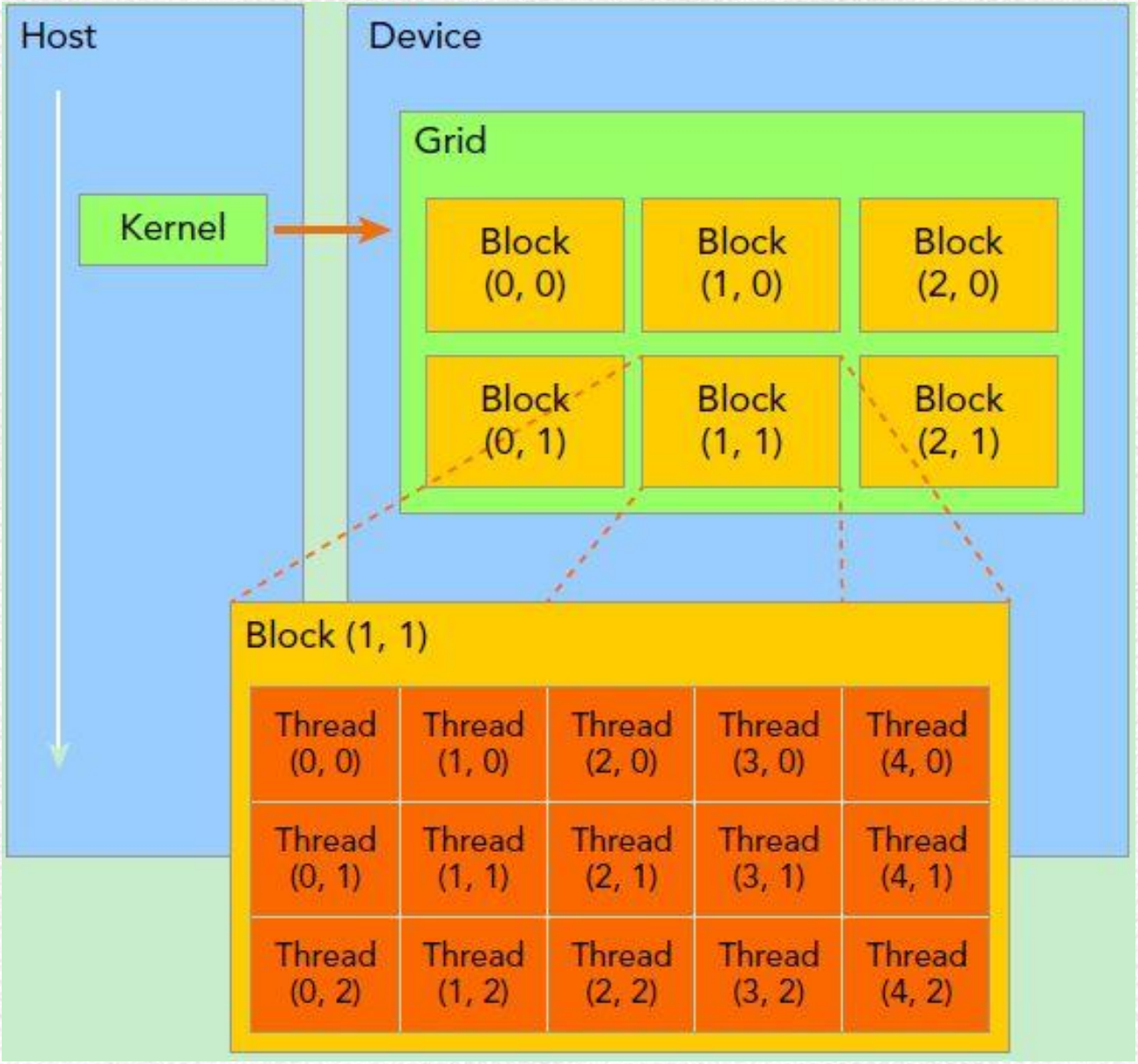
Grid, Block, Warp and Thread

Hui Liu
Email: hui.sc.liu@gmail.com

CUDA程序层次结构

- GPU上很多并行化的轻量级线程。
- kernel在device上执行时实际上是启动很多线程，一个kernel所启动的所有线程称为一个网格 (grid)
- 同一个网格上的线程共享相同的全局内存空间，grid是线程结构的第一层次
- 网格又可以分为很多线程块 (block), 一个线程块里面包含很多线程，这是第二个层次。
- warp: 32个线程一组, 这是第三个层次

CUDA 程序架构


Host
Device
Grid
Kernel
Block (0, 0)
Block (1, 0)
Block (2, 0)
Block (0, 1)
Block (1, 1)
Block (2, 1)
Block (1, 1)
Thread (0, 0)
Thread (1, 0)
Thread (2, 0)
Thread (3, 0)
Thread (4, 0)
Thread (0, 1)
Thread (1, 1)
Thread (2, 1)
Thread (3, 1)
Thread (4, 1)
Thread (0, 2)
Thread (1, 2)
Thread (2, 2)
Thread (3, 2)
Thread (4, 2)

CUDA程序层次结构

- grid 和 block 都是定义为dim3类型的变量
- dim3可以看成是包含三个无符号整数 (x, y, z) 成员的结构体变量，在定义时，缺省值初始化为1。
- grid和block可以灵活地定义为1-dim，2-dim以及3-dim结构
- 定义的grid和block如下所示，kernel在调用时也必须通过执行配置<<<grid, block>>>来指定kernel所使用的线程数及结构。
- 不同 GPU 架构, grid 和 block 的维度有限制

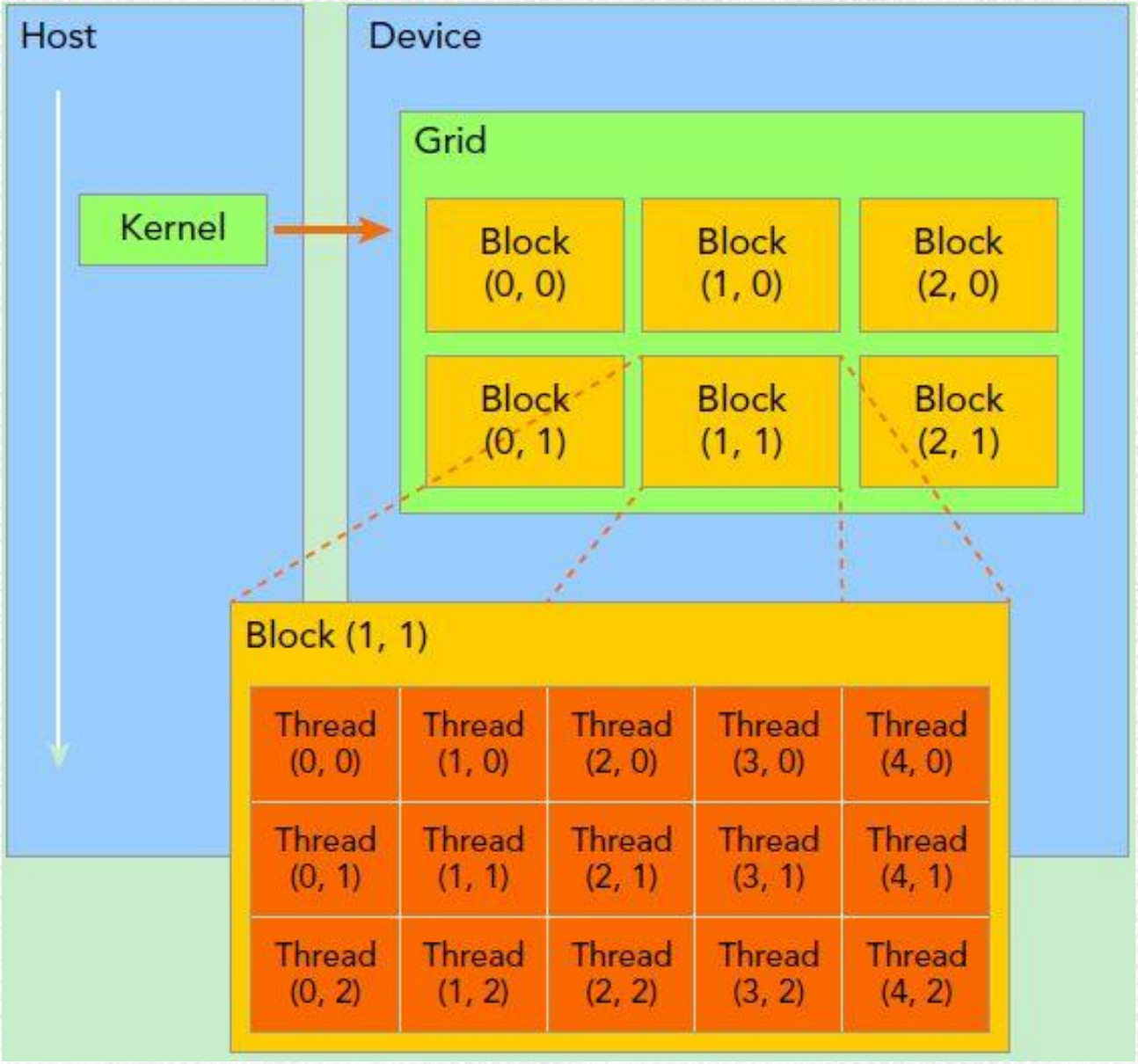
CUDA 程序调用

```
dim3 grid(3, 2);
dim3 block(5, 3);
kernel_fun<<< grid, block >>>(prams...);

dim3 grid(128,);
dim3 block(256);
kernel_fun<<< grid, block >>>(prams...);

dim3 grid(100, 120, 32);
dim3 block(16,16,4);
kernel_fun<<< grid, block >>>(prams...);
```

CUDA 程序架构


Host
Device
Grid
Kernel
Block (0, 0)
Block (1, 0)
Block (2, 0)
Block (0, 1)
Block (1, 1)
Block (2, 1)
Block (1, 1)
Thread (0, 0)
Thread (1, 0)
Thread (2, 0)
Thread (3, 0)
Thread (4, 0)
Thread (0, 1)
Thread (1, 1)
Thread (2, 1)
Thread (3, 1)
Thread (4, 1)
Thread (0, 2)
Thread (1, 2)
Thread (2, 2)
Thread (3, 2)
Thread (4, 2)

CUDA 内置变量

- 一个线程需要两个内置的坐标变量（blockIdx，threadIdx）来唯一标识，它们都是dim3类型变量，其中blockIdx指明线程所在grid中的位置，而threaIdx指明线程所在block中的位置:
- threadIdx包含三个值: threadIdx.x, threadIdx.y, threadIdx.z
- blockIdx同样包含三个值: blockIdx.x, blockIdx.y, blockIdx.z

- 逻辑顺序: X > Y > Z

dim3 grid(3, 2);
dim3 block(5, 3);

- 块: (0, 0), (1, 0), (2, 0), (0, 1), (1, 1), (2, 1)
- 线程: (0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (0, 2), (1, 2), (2, 2), (3, 2), (4, 2)

THANK YOU