cuda中threadIdx、blockIdx、blockDim和gridDim的使用

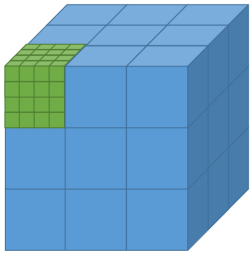
修仙

最浅显的理解和记录，方便后续学习查看，不保证结论正确性(;_;)

1.主要概念与名称

- **主机：**将CPU及系统的内存(内存条)称为主机;
- **设备：**将GPU及GPU本身的显示内存称为设备;
- **线程(Thread)：**一般通过GPU的一个核进行处理;
- **线程块(Block)：**由多个线程组成；各block是并行执行的，block间无法通信，也没有执行顺序。
- **线程格(Grid)：**由多个线程块组成。
- **核函数(Kernel)：**在GPU上执行的函数通常称为核函数;一般通过标识符__global__修饰，调用通过<<<参数1,参数2>>>，用于说明内核函数中的线程数量，以及线程是如何组织的。

画个图直观理解一下，下图是1个线程格，里面包含了9块线程块(蓝色的格子)，每个线程块里面又包含了16个线程(绿色的格子)。线程是最小的单位了，虽然这边我画的还是立方体，但通常是看做一个点



2.threadIdx、blockIdx、blockDim和gridDim

以上图为例子，把线程格和线程块都看作一个三维的矩阵。这里假设线程格是一个 `3*3*3` 的三维矩阵， 线程块是一个 `4*4*4` 的三维矩阵。

gridDim

所以有

`gridDim.x=3` `gridDim.y=3` `gridDim.z=3`

blockDim

`blockDim.x` 、 `blockDim.y` 、 `blockDim.z` 分别表示**线程块**中各个维度的大小，所以有

`blockDim.x=4` `blockDim.y=4` `blockDim.z=4`

blockIdx
blockIdx.x、blockIdx.y、blockIdx.z分别表示**当前线程块所处的线程格的坐标位置**

threadIdx
threadIdx.x、threadIdx.y、threadIdx.z分别表示**当前线程所处的线程块的坐标位置**

- 线程格里面总的线程个数N即可通过下面的公式算出

```
N = gridDim.x * gridDim.y * gridDim.z * blockDim.x * blockDim.y * blockDim.z
```

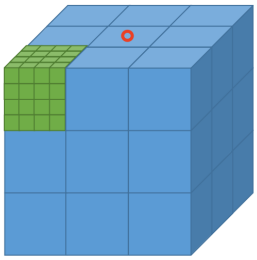
同时，通过 `blockIdx.x` 、 `blockIdx.y` 、 `blockIdx.z` 、 `threadIdx.x` 、 `threadIdx.y` 、 `threadIdx.z` 就可以完全定位一个线程的坐标位置了。具体看下面的**例子**：

将所有的线程排成一个序列，序列号为 $0,1,2,\cdots,N$ ，如何找到当前的序列号

1.先找到当前线程位于线程格中的哪一个线程块 `blockId`

```
blockId = blockIdx.x + blockIdx.y*gridDim.x + blockIdx.z*gridDim.x*gridDim.y;
```

说明一下这个算式的含义，不妨可以**从后往前看，类似一个搭积木的方式**。还是以刚刚上面画的那个图为例子，假设我们要找的block是红色标注的那块。



先是blockIdx.z*gridDim.x*gridDim.y，也就是从下往上包含了红色block所在的那一层的下面所有层。(也就是我涂成红色的这两层。)

下面就是blockIdx.y*gridDim.x，也就是我涂成橘黄色的这部分

最后再加上blockIdx.x，就到达了目标点～

2.接着找到当前线程位于线程块中的哪一个线程 `threadId` ，思路和上面找block一样

```
threadId = threadIdx.x + threadIdx.y*blockDim.x + threadIdx.z*blockDim.x*blockD
```

3.计算一个线程块中一共有多少个线程M

```
M = blockDim.x*blockDim.y*blockDim.z
```

4.求得当前的线程序列号 `idx`

```
idx = threadId + M*blockId;
```

3. `add2.cu` ,CUDA函数实现

先放代码，这里实现的功能是两个长度为n的tensor相加，每个block有1024个线程，一共有n/1024个block。

```
__global__ void add2_kernel(float* c,
                            const float* a,
                            const float* b,
                            int n) {
    for (int i = blockIdx.x * blockDim.x + threadIdx.x; \
            i < n; i += gridDim.x * blockDim.x) {
        c[i] = a[i] + b[i];
    }
}

void launch_add2(float* c,
                 const float* a,
                 const float* b,
                 int n) {
    dim3 grid((n + 1023) / 1024); #+1023是为了向上取整
    dim3 block(1024);
    add2_kernel<<<grid, block>>>(c, a, b, n);
}
```

经过上面的解释，代码应该容易理解一些了，但还是有些地方要说明一下。首先是这一句

```
for(int i = blockIdx.x * blockDim.x + threadIdx.x;  i < n; i += gridDim.x * blo
```

dx.x代表row上面的起
程的数目。这时候会

有疑问为什么要用 i += gridDim.x * blockDim.x，这样不是超出gpu范围了吗?

答案参考：CUDA编程中的gridDim and blockDim，也就是说，是将线程网格在一维数组上进行一到，“网格跨步循环”。

> 特别是，当 x 方向 (gridDim.x*blockDim.x) 中的总线程数小于我希望处理的数组的大小时，通常的做法是创建一个循环并让线程网格通过整个数组。在这种情况下，在处理一次循环迭代之后，每个线程必须移动到下一个未处理的位置，由 tid+=blockDim.x*gridDim.x; 给出。实际上，整个线程网格正在跳过一维数据数组，一次一个网格宽度。这个主题，有时称为“网格跨步循环”。

参考资料

概念:shuzhiduo.com/A/RnJWQnV...

例子参考: blog.csdn.net/qq_437151...

add2.cu: godweiyang：PyTorch自定义CUDA算子教程与运行时间分析

编辑于 2022-11-14 08:44 • IP 属地上海

CUDA Python PyTorch

写下你的评论...

4 条评论 默认 最新

lei
大佬你好，这个for循环在执行i += gridDim.x * blockDim.x之后就退出了吧，n的值是不是小于他吧，还是不太明白这个for循环添加的意义
2022-10-12 • IP 属地广东 回复 喜欢

修仙 作者 ▸ **lei**
你好，这个循环的意义是用来处理，n大于总线程数的情况的。例如，n=10，总线程数为4，所以需要循环使用这些线程。 具体的例子参考CUDA编程中的gridDim and blockDim中的第三大点～
2022-11-14 • IP 属地上海 回复 喜欢

lei ▸ **修仙**
好的大佬，谢谢，还没解决
2022-11-09 • IP 属地广东 回复 喜欢

展开其他 1 条回复 >

推荐阅读

cub库(三) block radix sort 方法

CUDA编程：grid和block设计指南（九）

go语言golang实现区 blockchain源代码