CUDA编程

都禁售了还学个嘚

CPU和GPU的区别：
CPU少数几个快速的计算核心，GPU拥有几百上千不那么快的计算核心
CPU主要用于数据缓存和流程控制，GPU主要用于计算
在CUDA编程中，CPU和主存被称为主机（Host），GPU被称为设备（Device）

GPU计算流程

1.分配host内存，并进行数据初始化；

2.分配device内存，并从host将数据拷贝到device上；

3.调用CUDA的核函数在device上完成指定的运算；

4.将device上的运算结果拷贝到host上；

5.释放device和host上分配的内存。

由于SM的基本执行单元是包含32个线程的线程束，所以block大小一般要设置为32的倍数

SM(streaming multiprocessor)则是GPU的核心，又称为GPU大核。它由一定数量的寄存器、一定数量的共享内存、常量内存的缓存、纹理内存和表面内存的缓存、L1缓存、线程束(thread warp)调度器、SP组成。

CUDA逻辑层：Thread->Thread Block->Grid

CUDA硬件层：CUDA Core->SM->Device

数组结构体（AoS）就是一个数组，每个元素都是一个结构体，而结构体数组（SoA）就是结构体中的成员是数组

并行编程范式，尤其是SIMD（单指令多数据）对SoA更友好。CUDA中普遍倾向于SoA因为这种内存访问可以有效地合并。

并行算法

1 规约算法

1.1 数组求和

参数	复杂度
运行时间	$T(n)=O(\log(n))$
处理器数量	$P(n)=n/2$
成本	$C(n)=O(n\cdot\log(n))$

CUDA中，县城对数据的连续读取效率高于其他方式，故一般选择连续线程取址

2 扫描算法

2.1 数组依次求和

Step complexity即步骤复杂度，完成一个工作需要多少步

Work complexity即工作复杂度，完成工作一共需要的工作量

2.2 全面顺序扫描算法

Inclusive Sequential Addition Scan

计算效率：$O(N)$

work complexity:$O(N)$
step complexity:$O(N)$

2.3 朴素全面扫描算法

Naive Inclusive Parallel Scan

Step complexity：$O(\log_2 n)$

Work complexity：$O(n^2)$

2.4 Blelloch并行扫描算法

巨量元素、大量处理器（核心）

step complexity：$O(\log N)$
work complexity：$O(N)$

2.5 Hills Steele并行扫描算法

大量元素、大量处理器（核心）

step complexity：$O(N\log N)$
work complexity：$O(\log N)$

3 统计直方图

直接并行会遇到：read-modify-write问题

有三种实现paralle histogram的方法：

1. atomics
2. per_thread histogram, then reduce
3. sort, then reduce by key