并行计算大作业

——并行优化矩阵乘法

朱桐 10175102111
周亦然 10175102207

2019.12.27

solving matrix mult on CUDA

- host vs device, distributed memory
- streamprocessors
- blocks, wraps, threads, shared memory

子任务：给定 i, j, 计算 C_{ij}，也就是并行最外面的两个 for

solving matrix mult on CUDA

- 轻量级线程，切换成本较低
- 基本可以平等划分任务
- 一个线程对应一个子任务（开 $n \times m$ 个线程）

shared memory

- 公式: $C_{ij} = \sum_{k=0}^{bm} A_{ik} \cdot B_{kj}$
- 对于内层的两个循环 j, k，发现公用 A_{ix}
- 使用 share memory 缓存 A

share memory

- block 内公用
- `block_size` 必须是 am 的因子，保证同块内使用相同的一行
- 别忘了 `__sychronize`

Fast IO: buffered input & output

- `fread, fwrite instead of printf, scanf`
- no need to flush every time we read/write a number

Occupancy: optimal size for the number of blocks and threads

- 以 wrap 为计算单元
- thread 中有 register
- block 中有 share memory
- 每个 multiprocessor 也有最大的 register 和 share memory 和 active threads
- 如果资源不够将无法派出所有的资源进行运算

Occupancy: optimal size for the number of blocks and threads

- 我们需要让最大比例的显卡工作
- 引入 Occupancy 机制
- 通过 nvidia 官网上的 Occupancy calculator 进行运算
- `nvcc --ptxas-options=-v` to nvcc 查看内核函数使用寄存器个数
- 使用

Bottleneck: IO

- GPU 计算矩阵瓶颈在于 IO
- 输出浮点数

asychronize IO