中山大學
SUN YAT-SEN UNIVERSITY

国家超级计算广州中心
NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

并行程序设计与算法（实验）

10-CUDA并行矩阵乘法

吴迪、刘学正

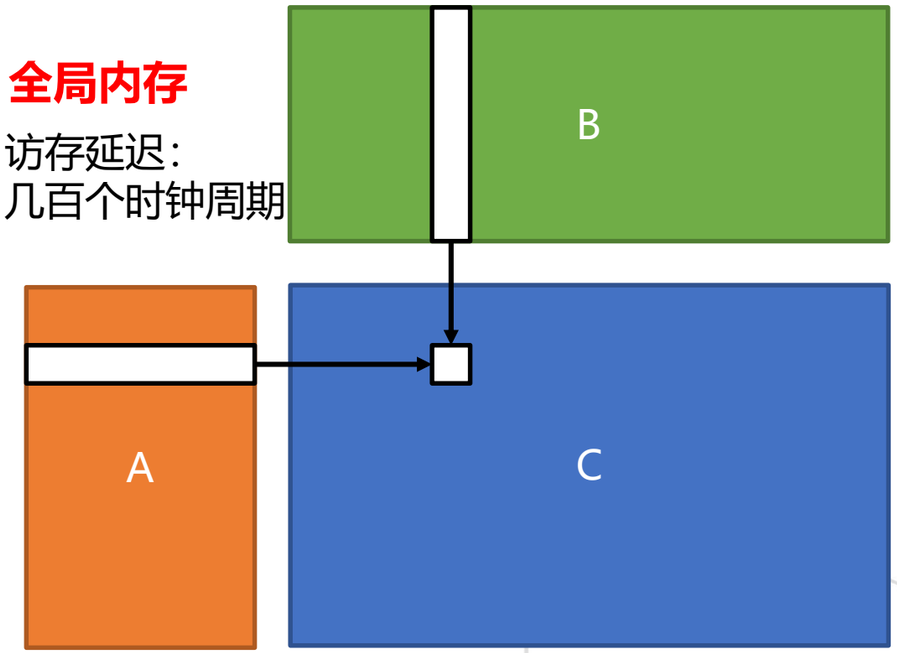
中山大学 计算机学院

实验概要

- 实验内容
  - 实现基础矩阵乘法
  - 优化矩阵乘法：共享内存，分块技术
  - 测量不同实现的运行时间
- 实验目的
  - 理解CUDA编程模型（Grid、Block、Thread）及其在矩阵乘法中的应用
  - 学习GPU内存优化技术

CUDA矩阵乘法朴素实现

- 原理：每个GPU线程计算输出矩阵的一个元素，通过遍历输入矩阵的行和列进行乘加运算，最终将结果写入全局内存
  - **全局内存带宽受限**：每个线程均要访问A和B的全局内存
  - **带宽浪费**：A（B）的同一行（列）被多个线程重复读取



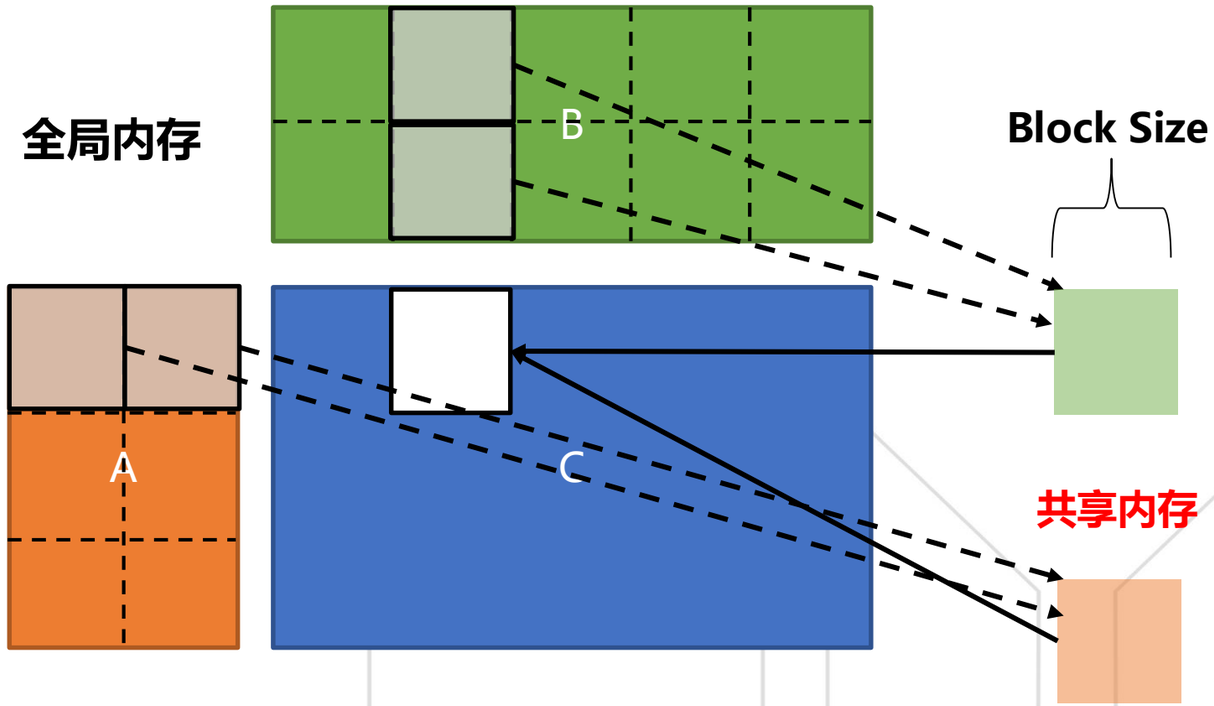
基本步骤：
1. 从全局内存读取A的一行
2. 从全局内存读取B的一列
3. 计算一次乘累加得到C的一个元素，并写回全局内存

```
__global__ void matrixMulKernelNaive(const float *d_A, const float *d_B, float *d_C, int m, int n, int k) {
    unsigned int row = blockIdx.y * blockDim.y + threadIdx.y;
    unsigned int col = blockIdx.x * blockDim.x + threadIdx.x;

    if (row < m && col < k) {
        float sum = 0.0f;
        for (int i = 0; i < n; i++) {
            sum += d_A[row * n + i] * d_B[i * k + col];
        }
        d_C[row * k + col] = sum;
    }
}
```

基于共享内存的实现优化

- 原理：将频繁访问的全局内存数据缓存在速度更快的共享内存中，通过分块计算实现数据重用，减少全局内存访问次数
  - **分块思想**：将大矩阵划分为小方块（block）
  - **数据重用**：A（B）每个block被所有计算同一行（列）的线程块重用

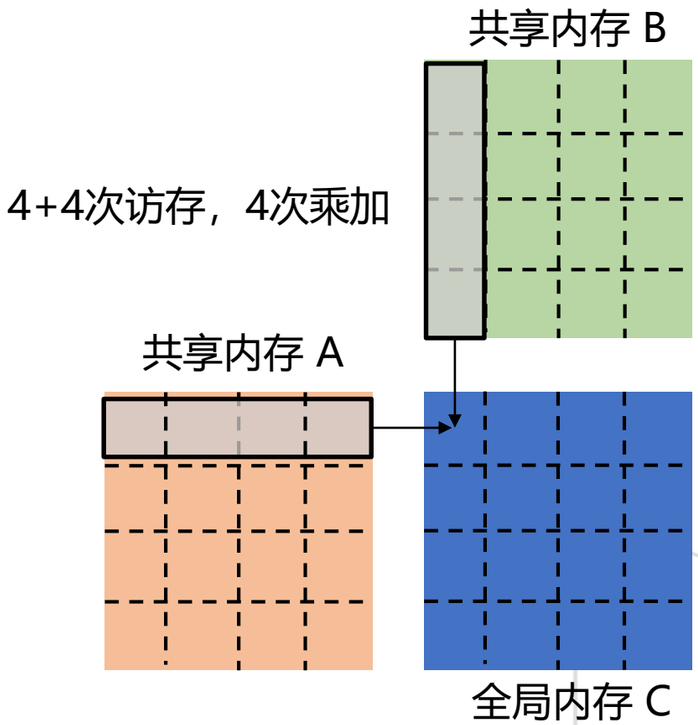


基本步骤：
1. 读取A矩阵一个tile到共享内存
2. 读取B矩阵一个tile到共享内存
3. 利用共享内存数据进行乘加运算
4. 写回一个block到全局内存

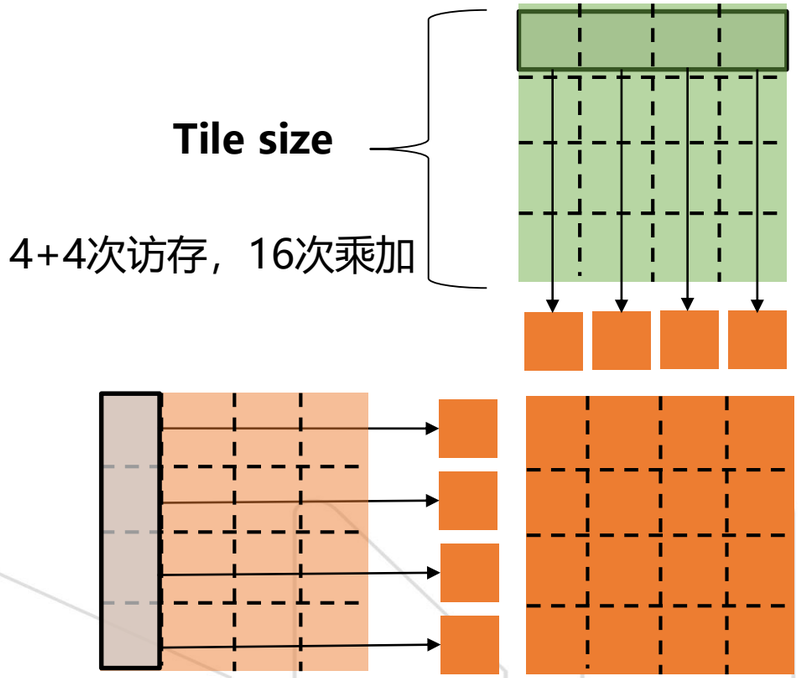
```
for (int j = 0; j < blockDim.x; j++) {
    sum += s_A[ty * blockDim.x + j] * s_B[j * blockDim.y + tx];
}
```

基于寄存器分块的优化

- 原理：将矩阵分块后，每个线程利用寄存器缓存子矩阵的部分数据，减少对共享内存的重复访问，提高计算访存比
  - **分块计算**：将大矩阵划分为小分块（Tile），每个线程处理一个Tile
  - **寄存器缓存**：线程将频繁使用的数据存储在寄存器中，避免重复加载



不使用寄存器分块优化



使用寄存器分块优化

基本步骤：

1. 从共享内存加载Tile Sizex1的A块到寄存器
2. 从共享内存加载1xtile_size的B块到寄存器
3. 利用寄存器数据进行乘加运算
4. 写回一个tile数据到全局内存

寄存器

访存延迟：
几个时钟周期

提示

- Tile Size通常小于或等于Block Size
- Tile Size决定寄存器个数
- Block Size决定共享内存的分块大小

实验内容

- 要求
  - 使用CUDA实现并行通用矩阵乘法
  - 分析不同因素性能的影响
    - 线程块大小、矩阵规模：如何提高占用率?
    - 访存方式：何时使用何种存储?
    - 数据/任务划分方式：按行，列，数据块划分等

Questions?