并行计算大作业报告

本章内容

Topic

- 实验内容概述
- 并行算法分析设计
- 实验数据分析
- 实验总结
- 上机实验与课程知识点分析

实验环境

实验内容概述

- 实验环境
- 实验1，2，3：天河超算
  - 网络：400GB/s
  - CPU：（单核双精度）9.2GFlops；（单节点双精度）588.8GFlops
  - GCC：版本9.3.0，gcc，g++，gfort
  - OpenMPI：版本4.1.1，mpicc，mpic++
- 实验4：异构计算平台
  - CPU：Hygon C86 7285 32-core Processor
  - 网络： 200Gb IB
  - 单卡性能：（FP64）10.1Tflops
  - HIP：版本5.4.23191
  - GCC：版本7.3.1，gcc，g++等

实验任务

实验内容概述

- 实验一
  - 熟悉使用实验环境进行实验，提交作业，查看队列，编写脚本文件
- 实验二
  - 使用pthread实现矩阵乘
- 实验三
  - 使用MPI实现矩阵乘
- 实验四
  - 异构计算使用DCU实现矩阵乘

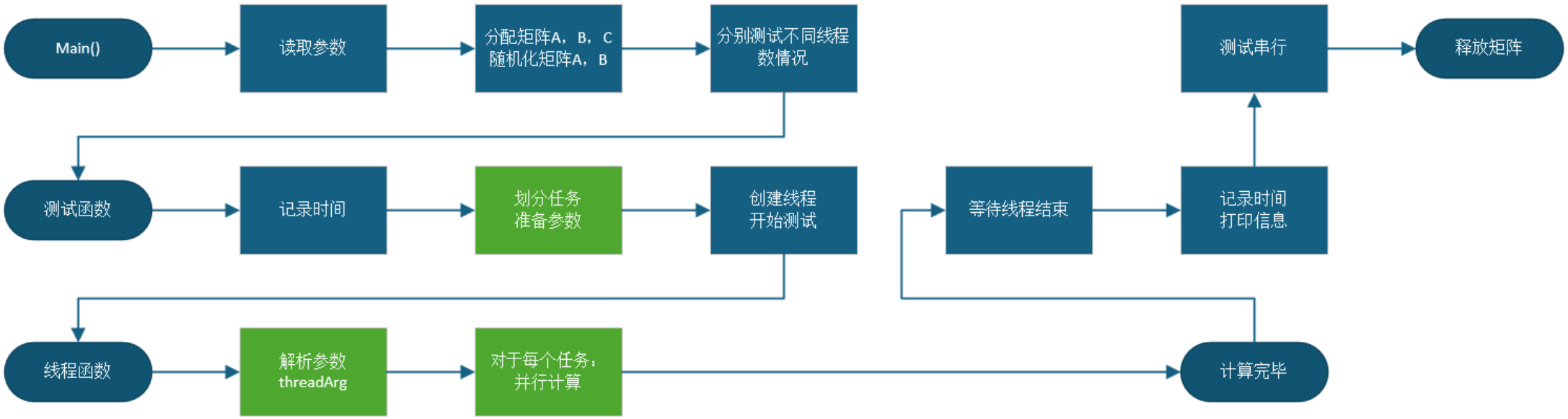
本章内容

Topic

- 实验内容概述
- 并行算法分析设计
- 实验数据分析
- 实验总结
- 上机实验与课程知识点分析

PThread多线程算法设计

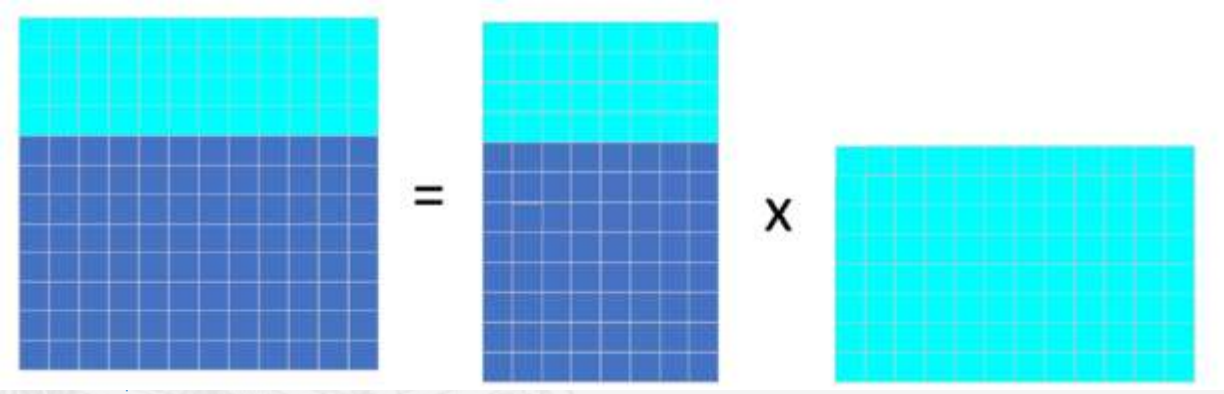
基本流程



PThread多线程算法设计

行划分

- 将输出矩阵C的行分配给不同线程，每个线程计算A的一部分行和B的所有列的乘积

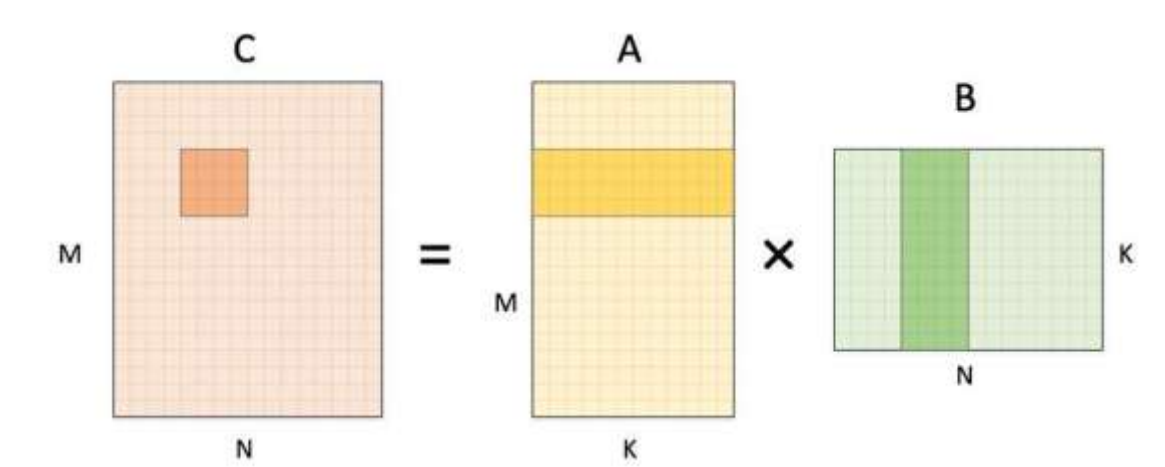



```
# 计算初始行区间
args.start_row ← base_length * j
args.end_row ← args.start_row + base_length - 1
# 边界条件处理：确保最后线程不越界
if j == num_threads - 1 then:
    args.end_row ← matrix_rows - 1
```

PThread多线程算法设计

块划分

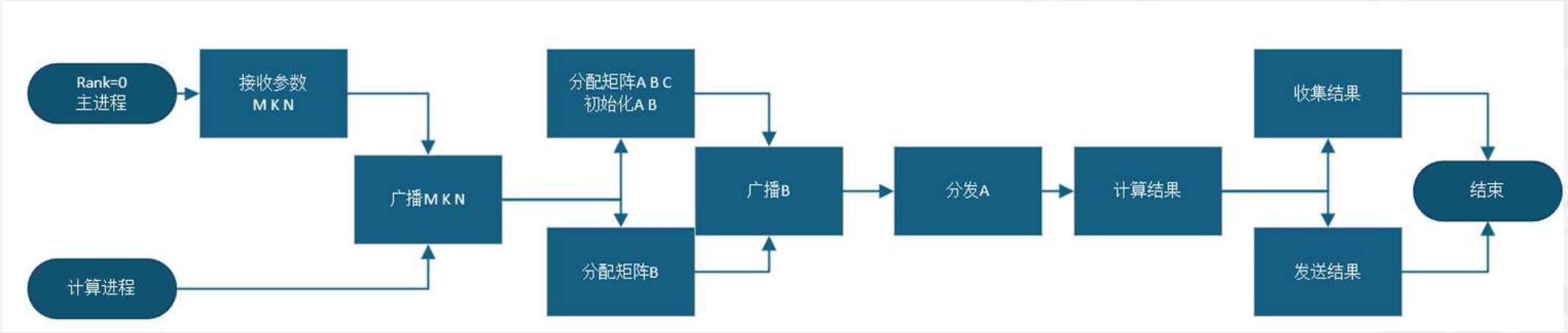
- 固定块的大小，将矩阵划分为子块，线程以块为单位计算局部结果




```
num_blocks_m = (矩阵行数M + BLOCK_SIZE - 1) // BLOCK_SIZE  # 行方向分块数
num_blocks_n = (矩阵列数N + BLOCK_SIZE - 1) // BLOCK_SIZE  # 列方向分块数
total_tasks = num_blocks_m * num_blocks_n                  # 总任务数=块矩阵元素总数

tasks_per_thread = total_tasks // n_threads                # 基础任务数
remainder = total_tasks % n_threads                        # 剩余任务分配

current_task = 0
for q in 0..n_threads-1:
    # 计算当前线程的任务区间
    end_task = current_task + tasks_per_thread + (q < remainder ? 1 : 0)

    # 设置线程参数
    thread_args[q] = {
        start_task: current_task,        # 起始任务编号
        end_task: end_task,              # 结束任务编号（不包含）
        num_blocks_n: num_blocks_n,      # 列方向块数（用于坐标转换）
        ...其他矩阵参数...
    }
```

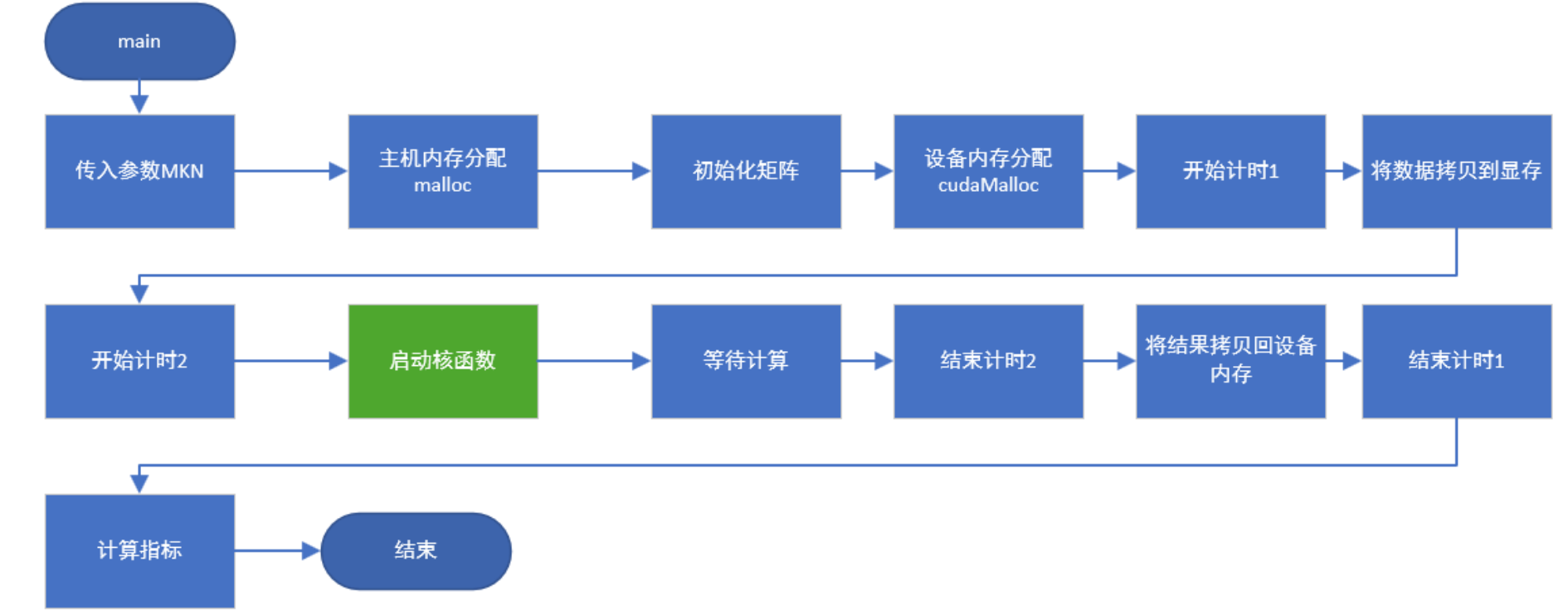
MPI并行算法设计

行划分



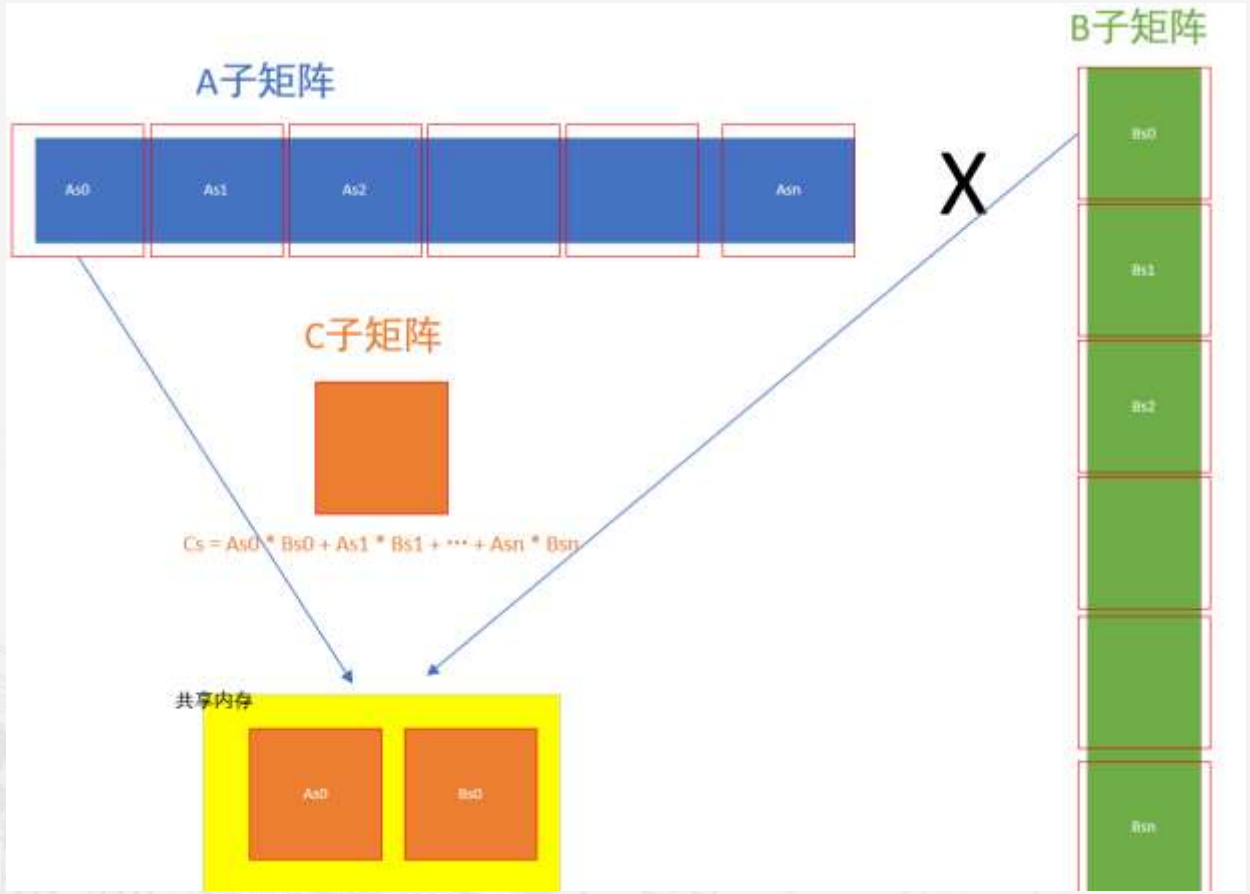
异构并行算法设计

流程图



异构并行算法设计

共享内存




```
for k_base in 0到K步长BLOCK_SIZE:
        # 阶段1: 协作加载A的分块到s_a ---------------------------
        a_row = by + ty          # 全局行坐标
        a_col = k_base + tx      # 全局列坐标
        if a_row < M 且 a_col < K:
            s_a[ty][tx] = A[a_row][a_col]  # 行优先加载
        else:
            s_a[ty][tx] = 0.0  # 边界填充
        # 阶段2: 协作加载B的分块到s_b ---------------------------
        b_row = k_base + ty      # 全局行坐标
        b_col = bx + tx          # 全局列坐标
        if b_row < K 且 b_col < N:
            s_b[ty][tx] = B[b_row][b_col]  # 行优先加载
        else:
            s_b[ty][tx] = 0.0  # 边界填充
        同步块内所有线程  # 等待数据加载完成
        # 阶段3: 共享内存矩阵乘累加 ----------------------------
        for kk in 0到BLOCK_SIZE-1:
            # s_a按行访问，s_b按列访问（矩阵乘法则）
            psum += s_a[ty][kk] * s_b[kk][tx]
```

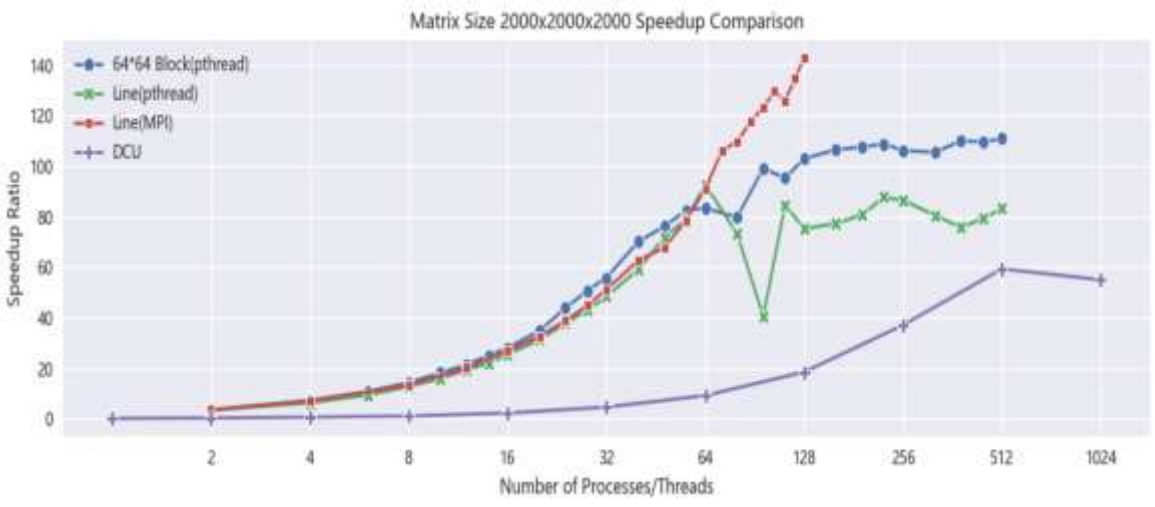
本章内容

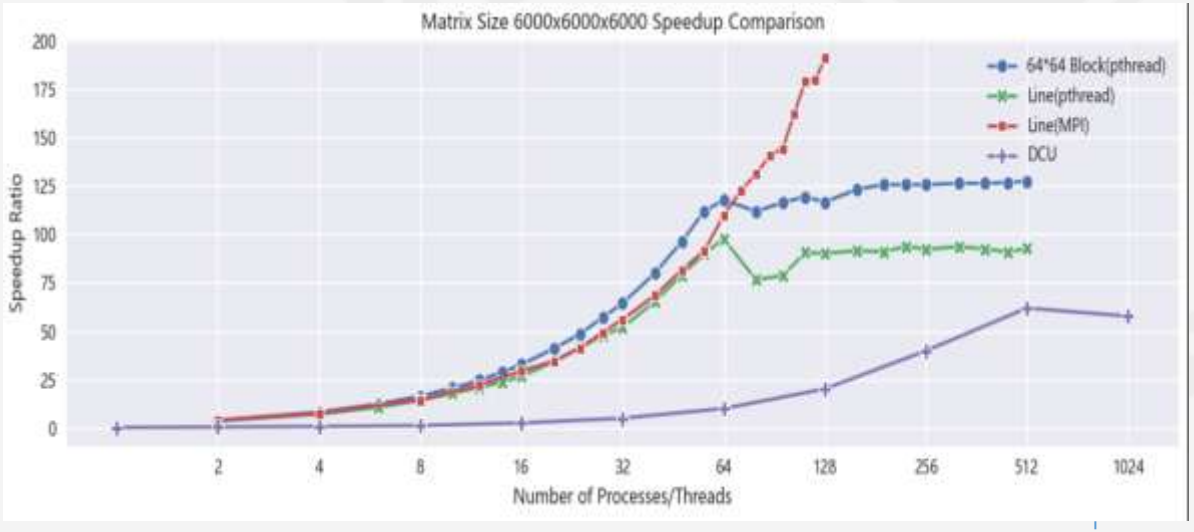
Topic

- 实验内容概述
- 并行算法分析设计
- 实验数据分析
- 实验总结
- 上机实验与课程知识点分析

实验数据综合分析

加速比对比





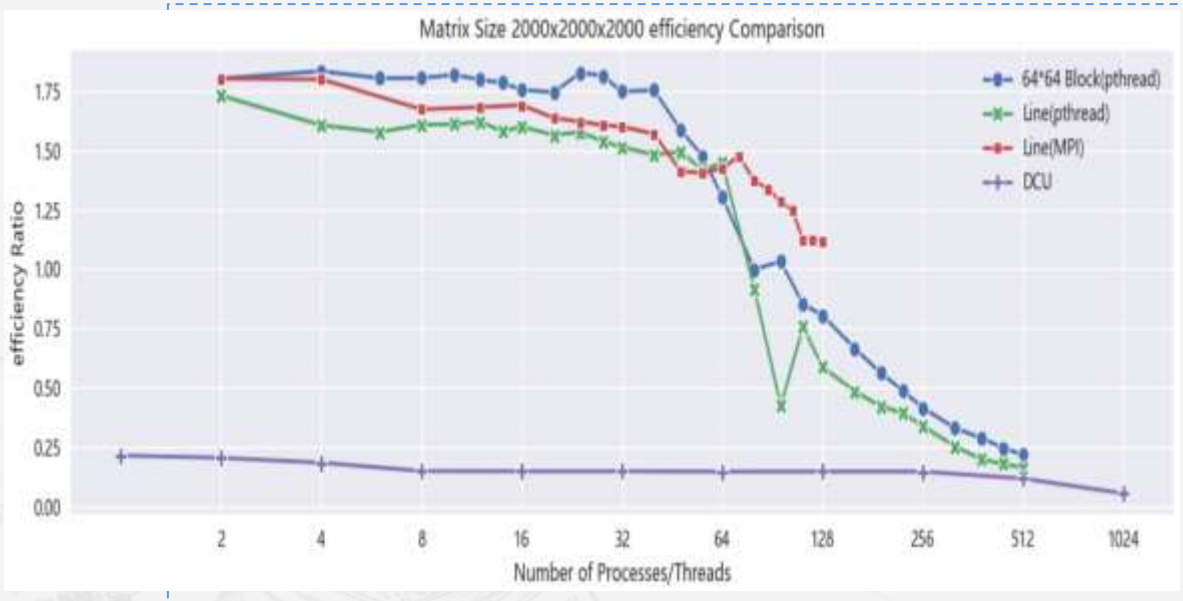
实验数据综合分析

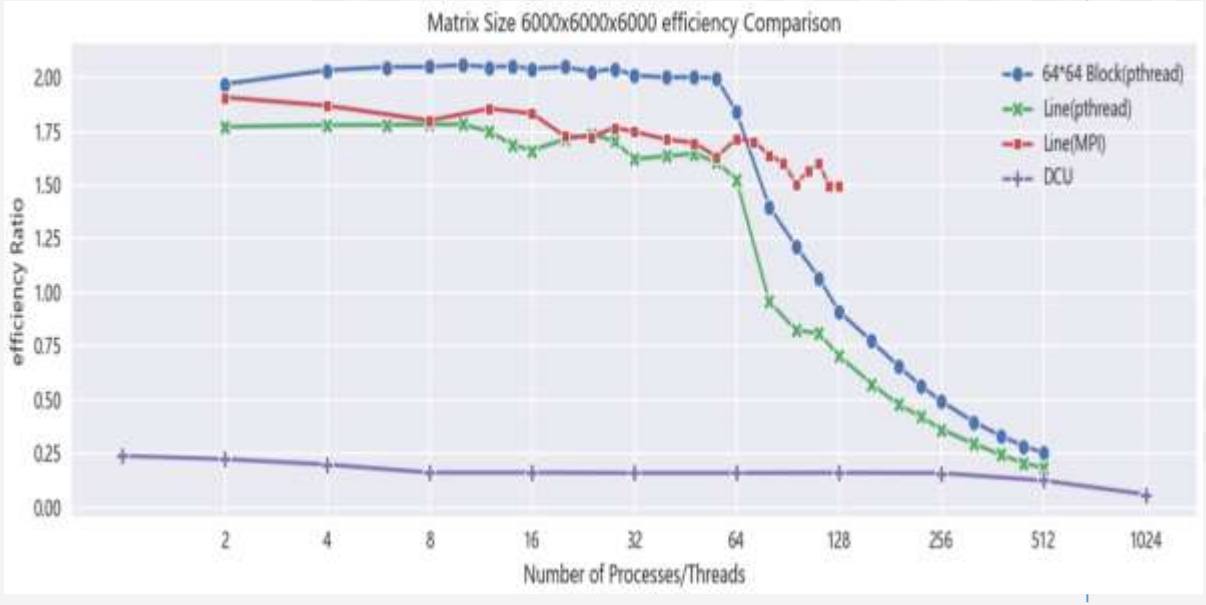
加速比对比

- 核数少于64：
  - 多线程块划分更优：开销小，局部性更优
  - 多进程与多线程行划分基本一致：单节点通信速度快
- 核数大于64：
  - 多线程：受限于单个物理节点，加速比无法继续提升
  - 多进程：跨节点通信，加速比可以继续提升
- 异构计算：线程开销极小，单个线程性能低，数量相同时，不如多进程多线程

实验数据综合分析

效率对比





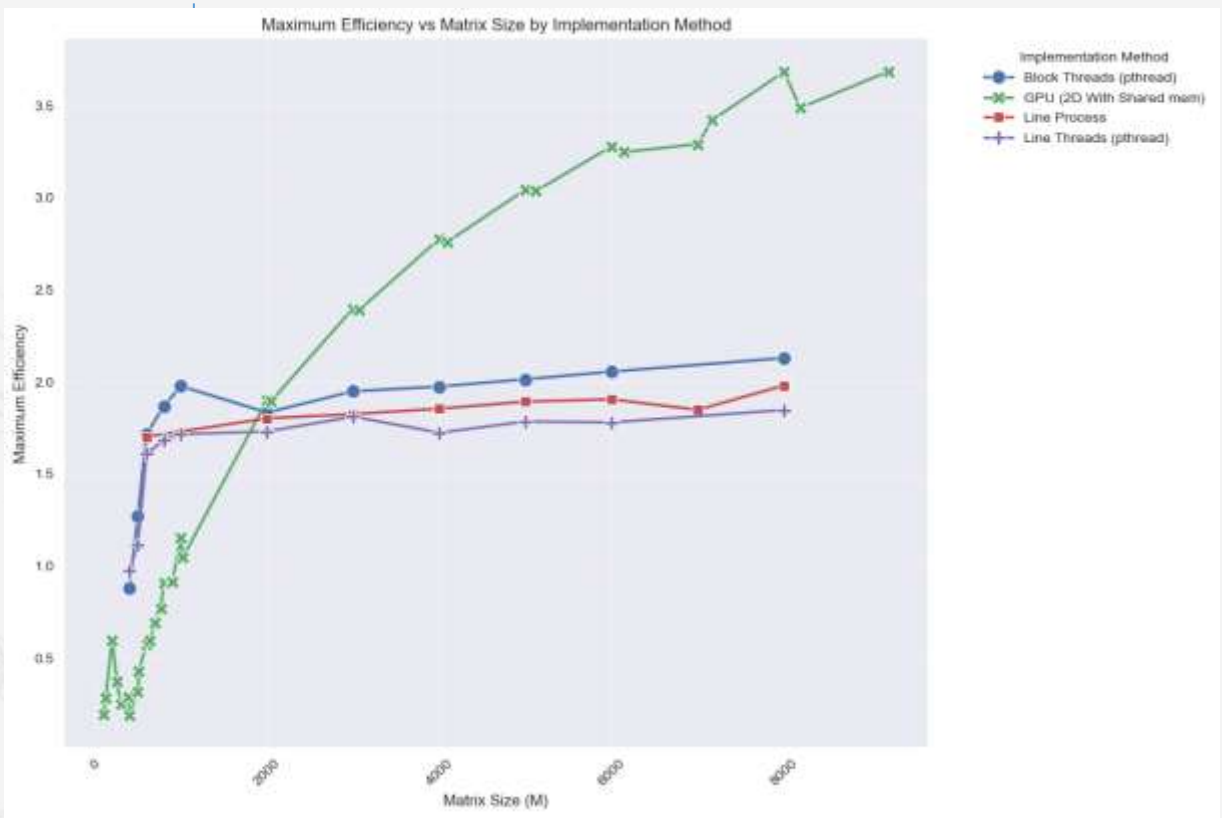
实验数据综合分析

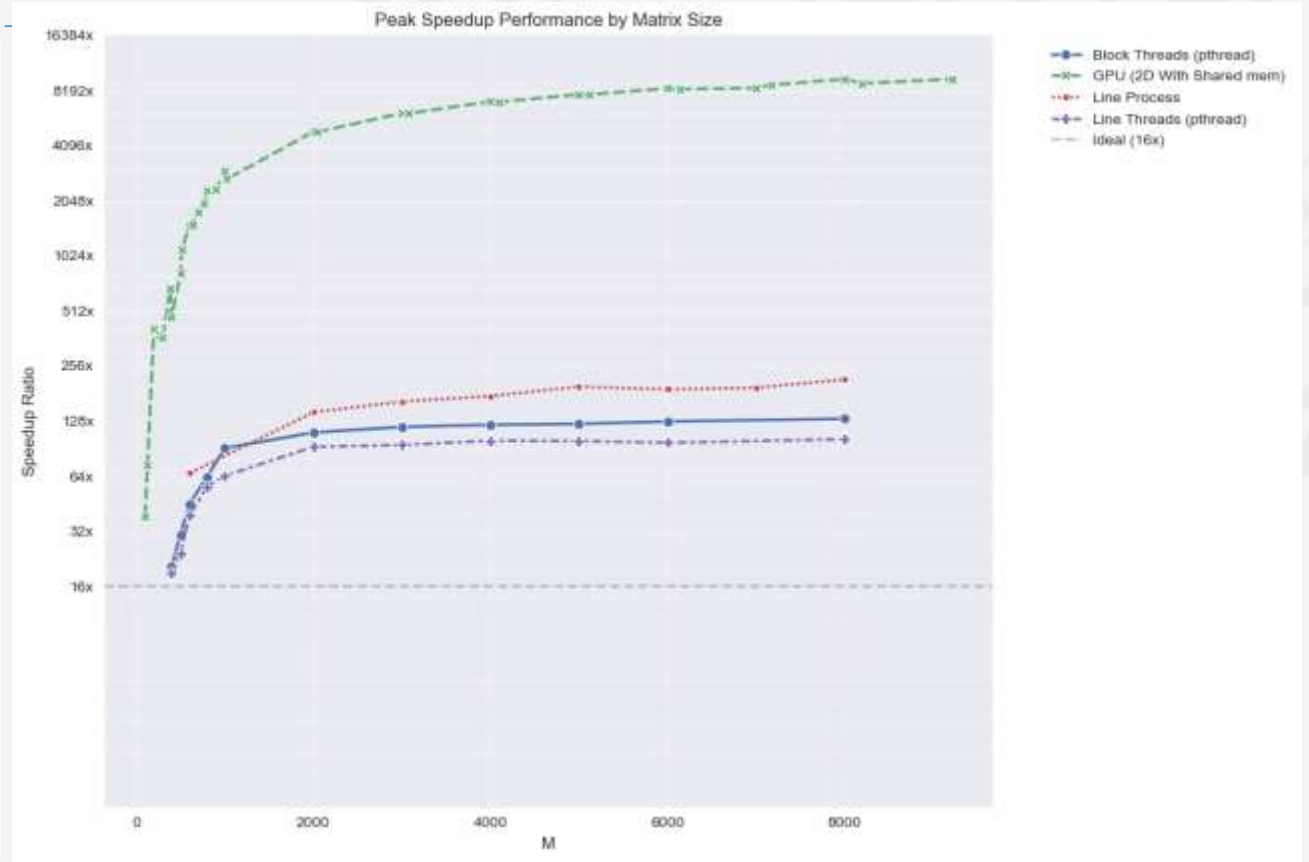
效率对比

- 核数少于64：
  - 多进程与多线程效率较平稳
- 核数大于64：
  - 多线程：加速比无法提升，效率急剧下降
  - 多进程：跨节点通信，虽然性能不及单节点内通信，效率有所下降，但优于多线程
- 异构计算：线程数相同，性能不及多进程，多线程(CPU)

实验数据综合分析

忽略进程/线程数





实验数据综合分析

忽略进程数/线程数

- 矩阵规模较小：
  - 加速比和效率较低，矩阵运算占比低，线程/进程总开销占比大。
- 矩阵规模增大：
  - 线程/进程开销比例降低，加速比和效率提高。
  - 规模大于一定数值时，保持相对稳定，可扩展性在一定程度上受到限制。
- DCU：
  - 特殊架构，适合大规模向量化计算

本章内容

Topic

- 实验内容概述
- 并行算法分析设计
- 实验数据分析
- 实验总结
- 上机实验与课程知识点分析

不同并行计算的分析

多线程

- 共享内存空间，通信高效，开销较低
- 优点：编程相对简单，通信延迟低，不需要显式实现，任务划分粒度可以自由调节，负载均衡实现起来容易。
- 缺点：仅限于单个进程内部，受限于物理机，可扩展性局限于单个节点内部增加cpu核数等，线程之间的同步较麻烦，一旦某个线程崩溃，会影响到整个的进程

不同并行计算的分析

多进程

- 独立内存空间，通过进程间通信交换数据
- 优点：可扩展性强
- 缺点：跨物理节点通信调试更加困难，编程难度较大，进程间负载均衡实现较为困难

不同并行计算的分析

异构计算

- 计算系统中同时包括多种不同架构的处理器，例如CPU，DCU，GPU，加速部件计算芯片占比大，控制芯片占比小，适合并行任务
  - 例如：8000*8000矩阵，32*32分块，每个block有32*32=1024个线程
  - 共(8000/32)* (8000/32)=62500个block，总计64000000个线程
- 优势：最大的发挥不同计算平台的优势
- 缺点：需要学习的架构较多，编程难度大

本章内容

Topic

- 实验内容概述
- 并行算法分析设计
- 实验数据分析
- 实验总结
- 上机实验与课程知识点分析

上机实验与课程知识点分析

实验2，3，4

- 数据分析部分：加速比，效率计算

加速比：串行执行时间与并行执行时间之比

$$S(n)=\frac{Execution\ time\ (one\ processor\ system)}{Execution\ time\ (multiprocessor\ system)}=\frac{t_s}{t_p}$$

效率

$$E=\frac{\text{Execution time using one processor}}{\text{Execution time using a multiprocessor} \times \text{number of processors}}$$

$$=\frac{t_s}{t_p \times n}=S(n)/n$$

上机实验与课程知识点分析

实验2，3，4

- 数据分析部分：加速比，效率计算
- 可扩放性的评测：
  - 计算机系统（或算法或程序等）性能随处理器数的增加而按比例提高的能力

加速比：串行执行时间与并行执行时间之比

$$S(n)=\frac{Execution\ time\ (one\ processor\ system)}{Execution\ time\ (multiprocessor\ system)}=\frac{t_s}{t_p}$$

效率

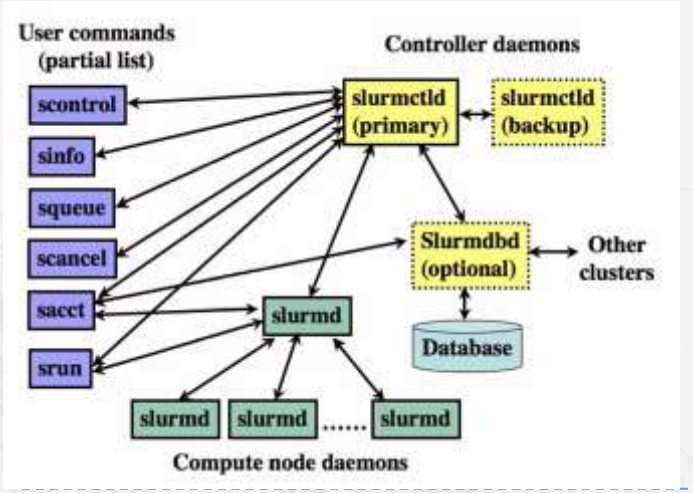
$$E=\frac{\text{Execution time using one processor}}{\text{Execution time using a multiprocessor} \times \text{number of processors}}$$

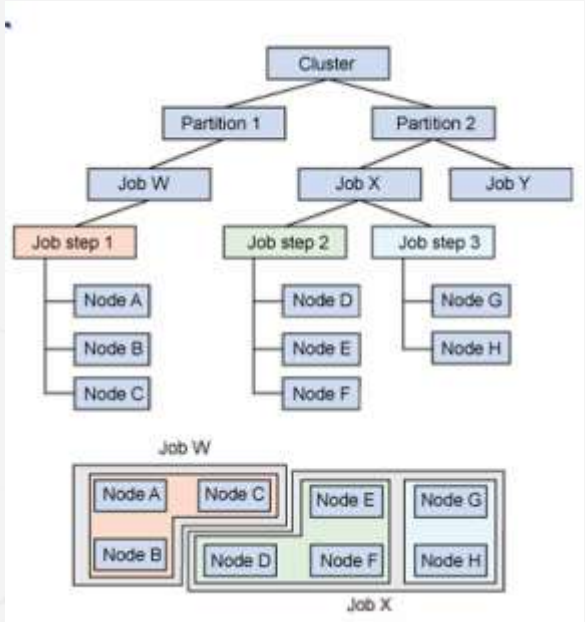
$$=\frac{t_s}{t_p \times n}=S(n)/n$$

上机实验与课程知识点分析

实验2，3，4

- 作业提交部分：slurm
  - 可用于大型计算节点集群的管理器和作业调度系统



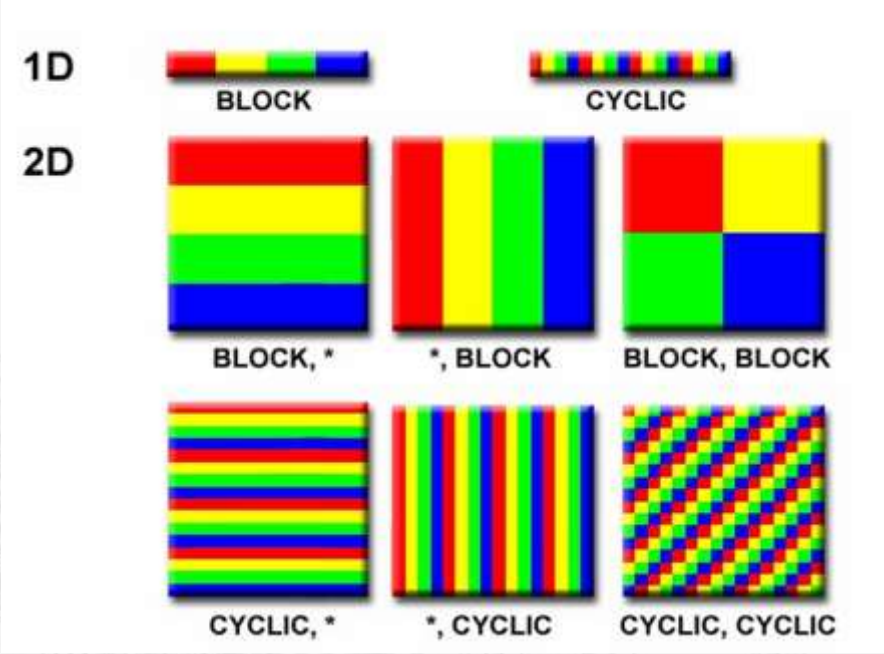


SLURM
#SBATCH -J name
#SBATCH -p cpu
#SBATCH --qos=debug
#SBATCH -t 5:00
#SBATCH -N 1
#SBATCH --cpus-per-task=4
#SBATCH --gres=gpu:1
#SBATCH -a 0-2
#SBATCH -o test.out
sbatch run.slurm
squeue
scancel 1234
salloc，手动切换
#SBATCH --nodelist=comput1

上机实验与课程知识点分析

实验2，3，4

- 任务划分部分：



上机实验与课程知识点分析

实验2

- 多线程基本概念
  - 调度，开销等
- 内存访问
  - 一致性：保证cache中数据与内存中数据相同的机制
  - 避免假共享
- Pthread
  - 线程函数，创建线程并执行，等待结束

上机实验与课程知识点分析

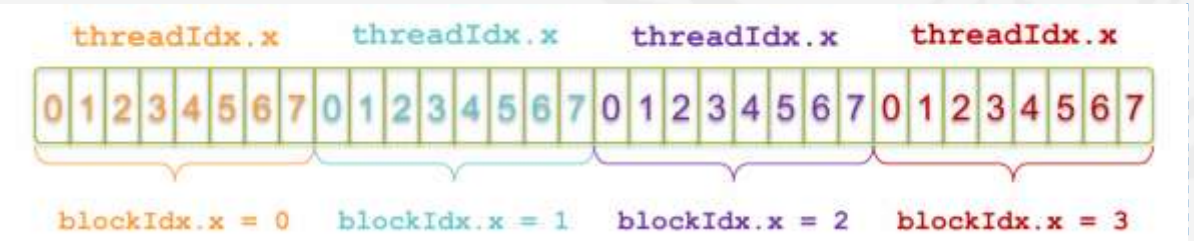
实验3

- MPI基本概念
  - 6个基本接口
  - 点对点通信：MPI_Send，MPI_Receive
  - 一对多：MPI_Bcast，MPI_Scatter
  - 多对一：MPI_Gather ，MPI_Reduce
- 阻塞通信
  - 通信模式：标准通信模式，缓存通信模式等
- 非阻塞通信：调用后可以立即返回
  - MPI_Isend，完成与检测
- 其他知识点：捆绑发送接收，自定义数据类型

上机实验与课程知识点分析

实验4

- 异构并行计算基本知识
  - 处理器架构设计依据任务具体特点优化
  - 最大程度发挥处理器性能
  - GPU计算过程：将数据从CPU内存拷贝到GPU显存，启动核函数计算，将结果拷贝回CPU
- CUDA编程示例，GPU向量加
- Blocks + Threads索引数组
- 线程层次：grid，block，thread



- 共享与同步：使用共享内存优化1D stencil，__syncthreads()同步
- 性能优化：不同于CPU，GPU/DCU环境中线程切换效率很高，使用尽可能多的线程数