1 实验内容

1.1 实验意义

矩阵乘法是科学计算的核心操作，优化其性能对深度学习、图像处理等领域至关重要。本实验通过 Pthread 多线程技术实现并行矩阵乘法，探究并行化对计算效率的提升。

1.2 相关工作

传统矩阵乘法的时间复杂度为 $O(N^3)$，通过并行化可将计算任务分配到多个线程，利用多核CPU资源提高性能。优化方向包括任务划分、缓存优化等。

2 实验方法

并行化设计可以通过按行划分矩阵 `A`，每个线程计算 `C` 矩阵的连续若干行，其中矩阵 `A`、`B`、`C` 为全局共享变量，每个线程保存起始行号 `start_row` 和结束行号 `end_row` 为本地变量.

代码分为四个函数，分别是串行矩阵乘法、线程计算、矩阵初始化、验证结果，主函数中初始化矩阵并调用串行和并行矩阵乘法函数，最后验证结果。

3 性能评估

3.1 实验环境

- 硬件配置
  - CPU: Intel(R) Core(TM) i5-12600KF（10核16线程：6P-core @4.9GHz, 4E-core @3.6GHz）
  - Cache 缓存：L1 80KB/核（P-core），L2 1.25MB/核（P-core），L3 20MB（共享）
  - 内存: DDR5 4800MHz（XMP: 6000MHz）32GB（双通道）
  - 主板: 微星B760M Gaming Plus WiFi D5
- 软件环境
  - OS: Ubuntu 20.04 LTS Desktop
  - 编译器: GCC 11.4.0，编译选项 -O3 -mavx2

3.2 性能对比

线程数	矩阵大小	串行时间(s)	并行时间(s)	加速比
1	512*512	2.34	2.35	1.00
2	512*512	2.34	1.18	1.98

线程数	矩阵大小	串行时间(s)	并行时间(s)	加速比
4	512*512	2.34	0.62	3.77
8	512*512	2.34	0.55	4.25

3.3 加速比分析

线程数增加时，加速比接近线性增长（如4线程加速比3.77），但受限于物理核心数（8线程加速比4.25）。

小矩阵（如32x32）因线程创建开销大，加速比低于1。

4 结论与优化措施

1. 结论
   并行化显著提升大规模矩阵乘法性能，加速比最高达4.25倍（8线程）。
   线程数超过物理核心数时，加速比提升有限。
2. 进一步优化
   分块技术（Tiling）：提升缓存命中率。
   SIMD指令：利用AVX2/AVX-512加速计算。
   内存布局优化：使用一维数组代替二维数组减少寻址开销。
   负载均衡：动态任务分配代替静态划分。
3. 体会
   通过本实验深入理解了多线程任务划分和性能瓶颈。实际优化中需平衡线程开销与计算负载，未来可结合硬件特性进一步优化