lab1

PB21051012 刘祥辉

算法设计

1.串行代码

矩阵LU分解采用Doolittle 算法

```
LU(Mat A):
    for k = 1 to n:
        for i = k + 1 to n:
            A[i][k] = A[i][k]/A[k][k]
        for i = k + 1 to n:
            for j = k + 1 to n:
                A[i][j] = A[i][j] - A[i][k] * A[k][j]
    Mat L, U
    for i = 1 to n:
        for j = 1 to n:
            if i > j:
                L[i][j] = A[i][j]
            else:
                U[i][j] = A[i][j]
        L[i][i] = 1
```

1.1 问题分析

并行化的部分:

- 外层的第一个for循环（k的循环）可以并行执行，因为每次迭代的操作都是相互独立的，不会有数据依赖性。
- 在第一个for循环内部的两个for循环（i的循环）可以在每次k迭代时并行执行，因为每次迭代的操作也是相互独立的。
- 第二个for循环内的两个嵌套循环（i和j的循环）也可以在每次k迭代时并行执行，因为它们之间也没有数据依赖性。

不可并行化的部分:

- 在生成L和U矩阵的部分，由于L的计算依赖于之前的迭代结果，所以不能并行化。同样，U的计算也具有依赖性，因此也不能并行化。

可能产生的空等:

- 在每次k迭代时，当执行内部的两个for循环时可能会产生空等待，因为第二个for循环依赖于第一个for循环的结果。在具体实现并行化时，需要确保这种情况下的线程同步，以避免空等待。

负载均衡划分:

- 由于每次迭代的工作量是随着矩阵大小而变化的，所以可能需要动态调整线程的分配，以确保负载均衡。这可以通过动态任务调度或者工作量划分来实现。

额外的并行化开销:

- 并行化可能会引入一些额外的开销，如线程创建、销毁和同步等。在设计并行化方案时，需要权衡这些开销与并行化带来的性能提升之间的关系。

1.2 算法描述

PCA框架

Problem (问题):

- 给定一个方阵A，需要计算其LU分解，即将A分解为下三角矩阵L和上三角矩阵U的乘积。

Approach (方法):

- 采用经典的LU分解方法。
- 利用并行计算来加速LU分解的过程。

Parallelism (并行性):

- 使用OpenMP来并行化计算过程，通过将内层循环并行化，实现对矩阵分解过程的加速。

Mapping (映射):

- 将每个线程映射到不同的矩阵元素上，通过并行计算来同时计算多个元素的值。

Implementation (实现):

- 通过在内层循环中使用OpenMP的并行指令，实现并行计算每个元素的值。
- 在计算U矩阵时，利用前面已经计算好的L和U的部分来加速计算过程。

伪代码

```
// 定义函数 Accumulate 用于计算矩阵乘法的累加值
函数 Accumulate(i, j, K):
    res = 0 // 初始化累加值为0
    对于 k 从 0 到 K-1 循环:
        res += L[i][k] * U[k][j] // 累加相乘结果
    返回 res // 返回累加值

// LU 分解过程
对于 i 从 0 到 N-1 循环: // 对每一行进行LU分解
    U[i][i] = A[i][i] - Accumulate(i, i, i) // 计算U矩阵的对角线元素
    L[i][i] = 1 // 设置L矩阵的对角线元素为1
    使用OpenMP并行化计算:
        对于 j 从 i+1 到 N-1 循环: // 对当前行的下一行进行操作
            U[i][j] = A[i][j] - Accumulate(i, j, i) // 计算U矩阵的非对角线元素
            L[j][i] = (A[j][i] - Accumulate(j, i, i)) / U[i][i] // 计算L矩阵的非对
角线元素
```

2.实验结果

强可扩放性分析，即在固定问题规模的情况下（OJ 平台），分析程序在不同核数下的性能表现。分析未能达到线性加速的可能因素。如果出现超线性加速，分析可能原因。

核数	时间/ms
1	6526ms
2	4682ms
4	3645ms
6	3312ms
8	3072ms

观察到随着核数的增加，程序的执行时间有所减少，但并未完全呈现出线性加速。这可能由以下因素导致：

1. **通信开销**：
    - 在并行计算过程中，可能存在线程间的通信开销，如数据传输、同步等操作，这些开销会随着核数增加而增加，影响了性能的线性扩展。
2. **负载不平衡**：
    - 某些任务可能无法等分给每个核，导致某些核的负载较重，而其他核的负载较轻。这会导致一些核完成任务后处于空闲状态，降低了整体性能。
3. **资源竞争**：
    - 在多核并行环境中，可能存在资源竞争，如共享内存的竞争、缓存争用等，这会导致性能下降。

3.总结

选好合适的算法进行并行化，不要只是并行已有的串行代码