《并行计算》实验报告（正文）

姓名 刘恒星 学名 2022229044 完成时间 2023-4-5

一、实验名称与内容

实验名称：多线程计算 PI 值

实验内容：

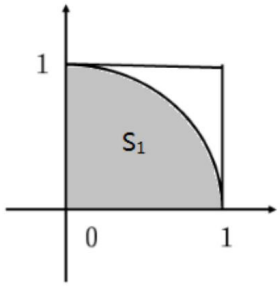
i. 积分法

$$\pi = \int_0^1 \frac{4}{1+x^2} dx \approx \sum_{0 \le i \le N} \frac{4}{1+(\frac{i+0.5}{N})^2} \times \frac{1}{N}$$

ii. 概率方法

如右图，在正方形中随机的投 n 个点，若有 m 个落入圆弧内，则：

$$\frac{m}{n} \approx \frac{S_1}{1} = \frac{\pi}{4}$$



二、实验环境的配置参数

CPU：国产自主 FT2000+@2.30GHz 56cores

节点数：5000

内存：128GB

网络：天河自主高速互联网络 400Gb/s

单核理论性能（双精度）：9.2GFlops

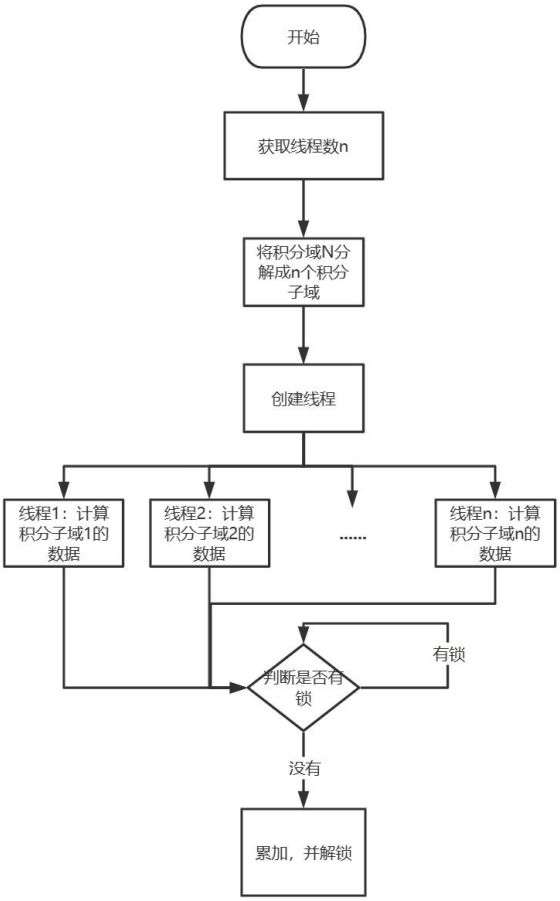
单节点理论性能（双精度）：588.8GFlops

三、实验题目问题分析

无论是积分法还是概率方法，都是遍历某一个数据域之后分别计算最后进行加和的计算方法。那么就可以从遍历数据域角度进行多线程设计，将数据域划分成若干个数据子域，每一个线程计算一个数据子域的结果，最后进行加和，从而实现并行化优化。

三、方案设计

积分法的设计思路：

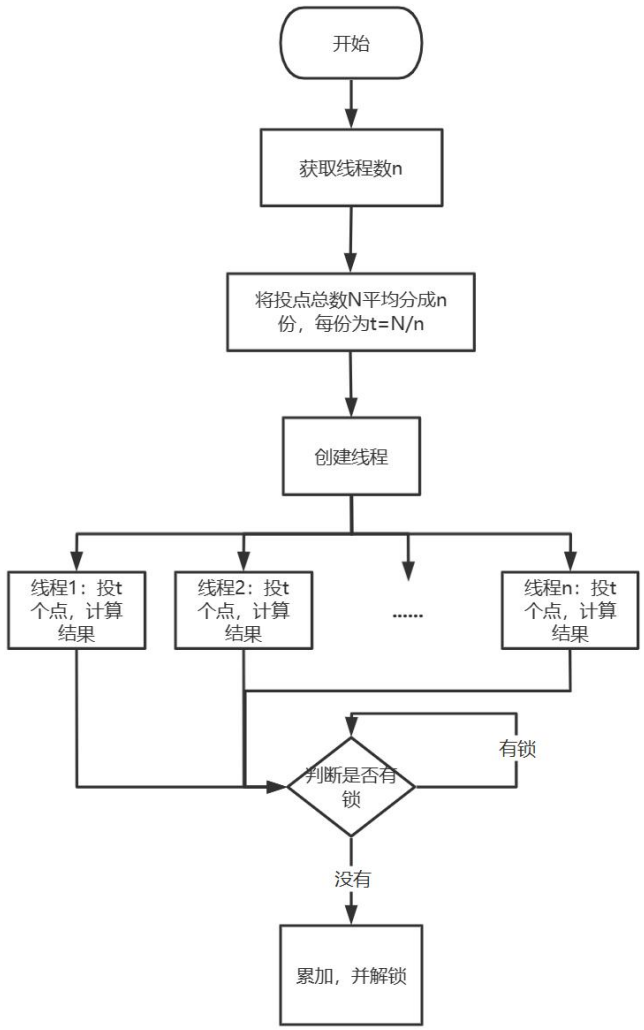


伪代码：

```
calculate_pi(id)
{
    len_per_thread = N / threadnum;

    st = id * len_per_thread;
    ed = (id + 1) * len_per_thread;
    thread_sum = 0;
    for i from st to ed:
        divided = 4;
        divisor = 1 + pow((i + 0.5) / n, 2);
        result = divided / divisor;
        thread_sum += result;
    pthread_mutex_lock(&mutex);
    sum += thread_sum;
    pthread_mutex_unlock(&mutex);
    return NULL;
}
for i from 0 to threadnum:
    pthread_create(&tid[i], NULL, calculate_pi, (void *)&i);
```

概率方法计算设计思路：



伪代码:

```
calculate_pi()
{
    int len_per_thread = N / threadnum;
    int thread_sum = 0;

    for i from 0 to len_per_thread:
        double dis = sqrt(x[i]*x[i] +  y[i]*y[i]);
        if dis <= 1:
            thread_sum ++;
    pthread_mutex_lock(&mutex);
    sum += thread_sum;
    pthread_mutex_unlock(&mutex);
    return NULL;
}
for i from 0 to threadnum:
    pthread_create(&tid[i], NULL, calculate_pi, (void *)&i);
```

五、实现方法

首先，在程序之中用常量定义总数 N，在命令行执行中加入指定的参数 n，这样程序就可以读取指定的参数进行数据域划分。随后，对于 n 个线程，使用 pthread_create 函数生成线程，并且让线程执行函数。两种方法分开说明

积分法：函数中积分通过参数 i 知道自己的数据子域的序号，从而计算出自己负责的数据子域的起点和终点，用对应的算法计算自己起点到重点的结果，随后用 pthread_mutex_lock 获得锁并加锁，最后获得锁加入到最后的结果中，用 pthread_mutex_unlock 解锁。

概率法：每一个线程都只需要模拟投出 t 个点，对于每一个点，计算在不在圆周范围内，从而算出这个线程下的结果，随后用 pthread_mutex_lock 获得锁并加锁，最后获得锁加入到最后的结果中，用 pthread_mutex_unlock 解锁。

线程执行结束之后，使用 pthread_join 函数终止线程并回收。

六、结果分析

本次实验的数据计算总数 N = 1e7

积分法：

并行的结果是：3.141592653590

串行的结果是：3.141592653590

结果正确。

本次实验并行采用了多种方案，分别使用了 2 线程，4 线程，8 线程进行实验。

串行程序运行时间为：0.596131s

2 线程并行程序运行时间： 0.298661s，加速比：1.996， 效率=0.998

4 线程并行程序运行时间： 0.149317s，加速比：3.992， 效率=0.998

8 线程并行程序运行时间： 0.095295s，加速比：6.255， 效率=0.781

概率法：

并行的结果是：3.141592653590

串行的结果是：3.141592653590

结果正确。

串行程序运行时间为：0.596551s

2 线程并行程序运行时间： 0.298592s，加速比：1.997， 效率=0.998

4 线程并行程序运行时间： 0.149305s，加速比：3.995， 效率=0.998

8 线程并行程序运行时间： 0.085023s，加速比：7.016， 效率=0.877

实验分析：可以发现，随着线程数量的增多，程序运行的速度显著减少，加速比增加，说明多线程并行优化确实可以加快程序速度。但是也可以发现，随着线程数量增加，效率也开始逐渐降低，因为多线程带来的频繁上下文切换也会带来额外的开销。

七、个人总结

通过这次实验，明白了如何使用 pthread 库实现多线程编程，了解了并行程序设计。从这次实验遇到的困难集中在对 pthread 的不熟悉，以及如何设计并行优化上。这次实验也让我明白了 pthread 库的初步使用和一些简单的并行程序的设计。通过实验结果，可以发现线程并不是越多越好，线程的增加会引起效率的降低，不加思考的引入线程会导致额外的开销，如何在效率和加速比中得到权衡是一个值得思考的问题。