并行计算课程

结 题 报 告

报告名称：	多线程计算π值不同计算方式的性能分析
姓　　名：	李润泽
学　　号：	3019244266
联系电话：	15942643201
电子邮箱：	Lirz3019244266@163.com
填写日期：	2021 年 3 月 26 日

2020 年制

一、实验内容概述

本实验要求通过多线程来计算π的数值来进行不同计算方式以及不同线程数的性能分析。目的为提升学生对并行计算的理解认识，培养学生编写基本并行程序的能力，加深对多线程（pthread）并行编程的理解和认识。

在计算π的过程中，共计三种计算方式，实验要求学生均需要编写出相应的代码，并通过改变线程数，分别进行三种方式的数据统计、计算时间记录以及性能分析，代码中 N 尽可能大，这样可以使计算值逼近真实值（尤其针对蒙特卡洛算法）。

1.计算方法：积分法、概率法（蒙特卡洛算法）以及幂级数计算方法

2.编程语言：C 或 C++

3.并行计算操作系统：天津大学超算平台 CentOS 7.6

4.编译环境：Intel 19.1.0.166

5.脚本编写：系统提交需要编写 PBS 脚本实现

6.数据分析要求：提供实验结果数据、加速比曲线以及效率

二、并行算法分析设计

（一）实现方法

1 输入线程数 t，与总数据量 N（程序定义 N=500000000）记录时间

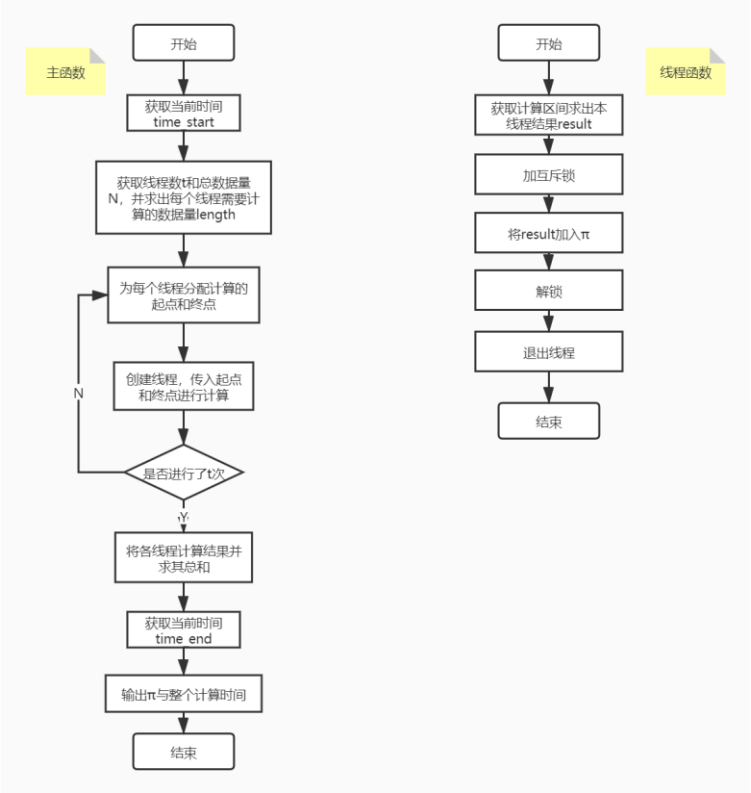
2 为每个线程分配计算的数据量

3 创建线程，每个线程计算各自部分

4 计算完成后，将各自线程计算的部分合起来求出最后结果，并再次记录时间

5 输出求出的π

（二）程序流程图



（三）算法

1 积分法

1.1 实验数学计算模型

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

1.2 代码（由于三种算法的主函数基本相同，故报告中只提供线程函数）

```
void *thread(void * ID){
    int id    = *(int*) ID;
    int i;

    for (i = id * length;i < (id + 1) * length && i < N ; i++){
        result[id] += 4.0/((1 + ((i + 0.5)/N) * ((i + 0.5)/N)));
    }

    pthread_mutex_lock (&mutex);
     PI += result[id] ;
     pthread_mutex_unlock (&mutex);
```

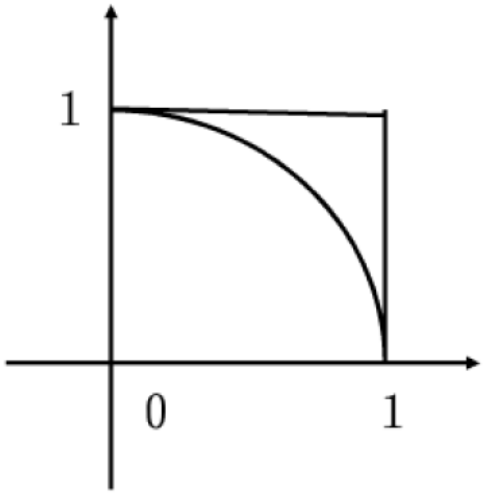
}

2 概率法（蒙特卡洛算法）

2.1 实验数学计算模型

在正方形中随机的投 n 个点，若有 m 个落入圆弧内，则：

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



2.2 代码

```
void *thread_function(void *ID){
    int id=*(int*)ID;
    int i;
    double x,y;
    srand((int)time(NULL));

    for(i=0; i<length; i++){
        x=1.0*rand()/RAND_MAX;
        y=1.0*rand()/RAND_MAX;
        if(x*x+y*y<=1.0) result[id]++;
    }

    pthread_mutex_lock(&mutex);
    pi+=result[id];
    pthread_mutex_unlock(&mutex);

}
```

3 幂级数法

3.1 实验数学计算模型

计算公式：$\pi = 4 \times \arctan(1) = 4 \times (1 - \frac{1}{3} + \frac{1}{5} - \cdots + \frac{(-1)^{n+1}}{2n-1} - \cdots)$

3.2 代码

```
void *thread(void *ID){
    int id=*(int*)ID;
    int i;
    for(i=id*length; i<(id+1)*length && i<N; i++){
        if(i%2==0){
            result[id]+=(1.0/(i*2+1));
        }
        else{
            result[id]-=(1.0/(i*2+1));
        }
    }

    pthread_mutex_lock(&mutex);
    pi+=result[id];
    pthread_mutex_unlock(&mutex);
}
```

（四）脚本代码（以积分法脚本、线程数 32 为例）

```
#!/bin/bash
#PBS -N test
#PBS -q qstudent
#PBS -l nodes=1:ppn=32
#PBS -j oe

#cd $PBS_O_WORKDIR
echo " ">>pi_1.log
echo "ppn=32">>pi_1.log

date +%s.%N >>pi_1.log
./pi_i 32 >>pi_1.log
```

```
date +%s.%N >>pi_1.log
echo " ">>pi_1.log
```

三、实验数据分析

（一）实验环境

CPU：Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz　　2.59 GHz

内存：16.0GB

互联网络参数：172.23.80.20（用的是校园网）

（二）实验数据综合分析

加速比：$S(n)=\frac{\text{单线程计算时间}}{\text{多线程计算时间}}=\frac{t_s}{t_p}$

效率：$E=\frac{\text{单线程计算时间}}{\text{多线程计算时间}\times\text{处理器数}}=\frac{t_s}{t_p\times n}$

1 实验数据

1.1 积分法

ppn	PI	run_time	average_run_time	speedup	efficiency
1	3.14159265	7.691020012	7.68925333	1.000000	1.000000
		7.683599949			
		7.69314003			
2		12.27024007	13.58782999	0.565893	0.282946
		16.19499993			
		12.29824996			
4		36.30519986	35.62618661	0.215832	0.053958
		36.26209998			
		34.31125998			
8		6.533760071	6.902730068	1.113944	0.139243
		6.750659943			
		7.423770189			
16		8.556809902	9.779683352	0.786248	0.049140
		12.15327001			
		8.628970146			

32		4.770779848	4.753093322	1.617737	0.050554
		4.900130033			
		4.588370085			

1.2 概率法（蒙特卡洛算法）

ppn	**PI**	**run_time**	**average_run_time**	**speedup**	**efficiency**
1	3.14150087	29.144500017	29.163783312	1.000000	1.000000
	3.14151415	29.180539846			
	3.14160567	29.166310072			
2	3.14155057	263.383389950	335.662976583	0.086884	0.043442
	3.14162139	264.268469810			
	3.14158016	479.337069988			
4	3.14158905	153.655009985	158.449386676	0.184057	0.046014
	3.14152603	161.798300028			
	3.14150011	159.894850016			
8	3.14152510	391.214679956	352.894660076	0.082642	0.010330
	3.14173570	383.270110130			
	3.14170284	284.199190140			
16	3.14156070	360.141720057	371.864013354	0.078426	0.004902
	3.1416320	376.71350002			

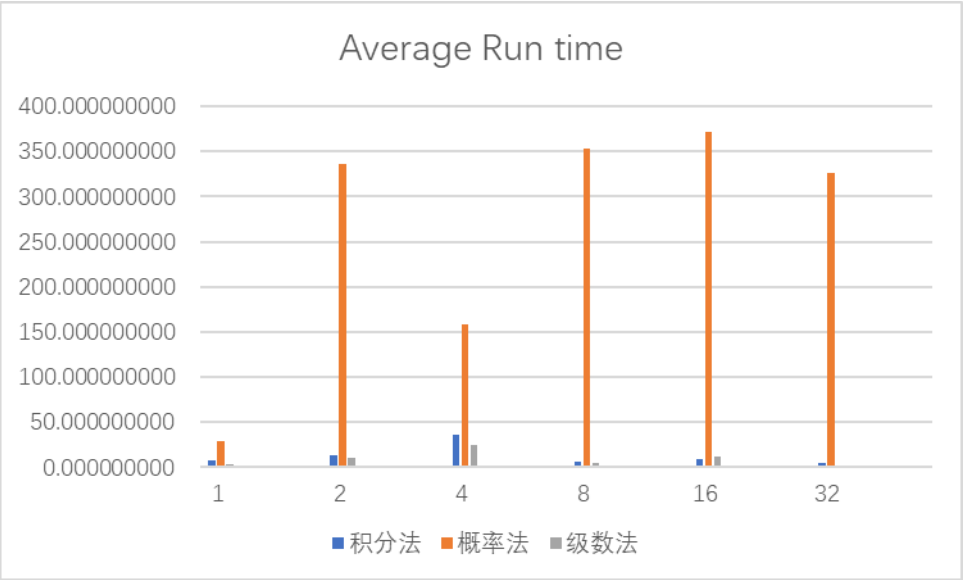
	2	3			
	3.14157079	378.736819983			
32	3.14170041	326.024610043	326.228766680	0.089397	0.002794
	3.14162449	332.068710089			
	3.14160957	320.592979908			

1.3 幂级数法

ppn	**PI**	**run_time**	**average_run_time**	**speedup**	**efficiency**
1	3.14159265	4.032389879	4.027999957	1.000000	1.000000
		4.024209976			
		4.027400017			
2		11.527930021	10.022126675	0.401911	0.200955
		11.947000027			
		6.591449976			
4		30.654780149	25.106626749	0.160436	0.040109
		10.267790079			
		34.397310019			
8		5.978379965	4.759400050	0.846325	0.105791
		3.501110077			
		4.798710108			
16		12.593540192	12.059343338	0.334015	0.020876
		12.452999830			
		11.131489992			
32		1.737519979	2.704423348	1.489412	0.046544
		3.266760111			
		3.108989954			

2 性能分析

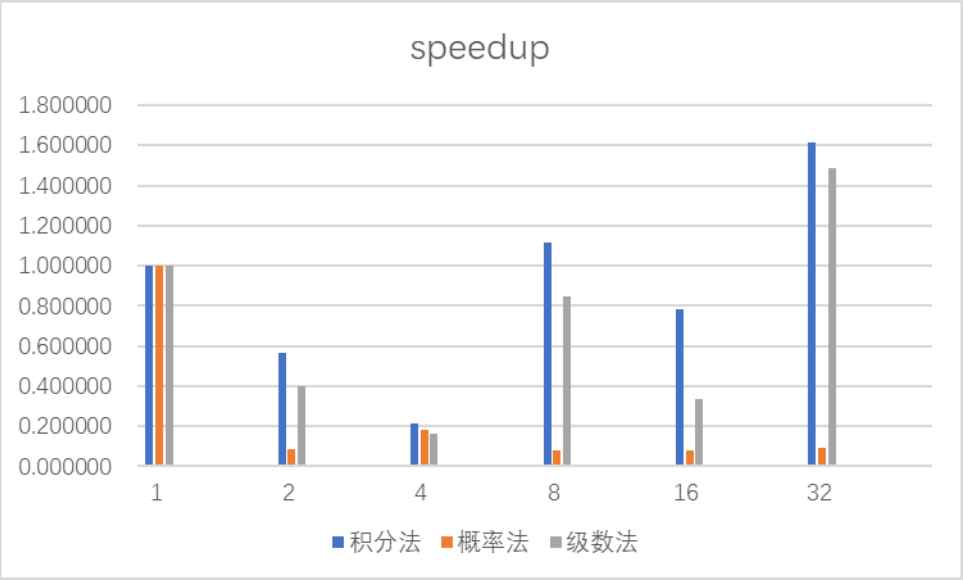
2.1 平均运行时间



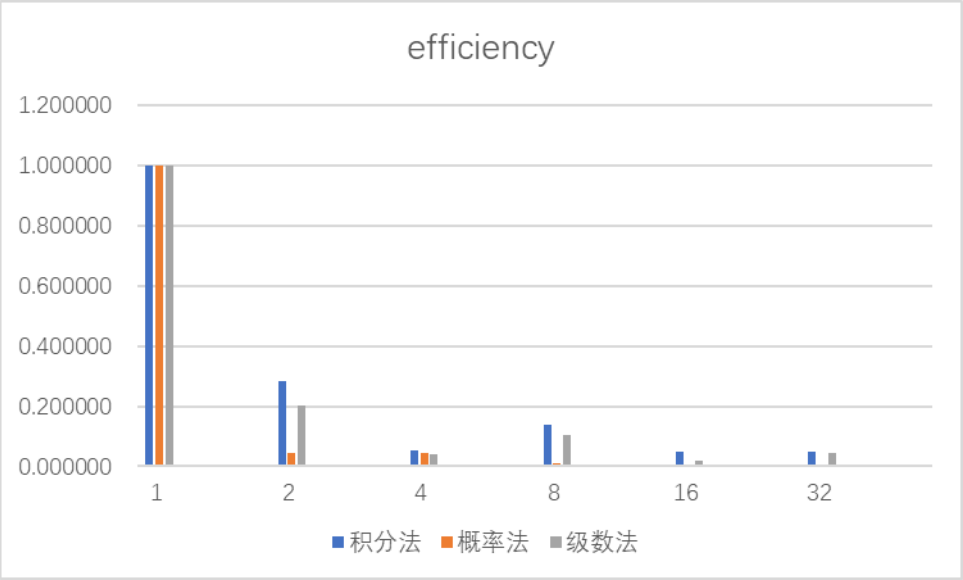
平均运行时间是三次运行取的平均值，结果相对可靠。大体可以看出，在三种方法中，运行时间随着线程数的增加会不断上升，之后会逐渐下降。因此并不是线程越多运行时间越快。

此外，概率法的运行时间最长，可能是因为每次循环程序都要取随机数，调用<time.h>头文件，因此时间最长。

2.2 加速比



2.3 效率



四、实验总结

根据分析，我们可以看出：

1.在线程数量较小时，多线程反而会比单线程时间更长，但在线程数量较大的时候可以得出相对理想的效果。得出结论：大体而言，多线程对大规模数据效果更好。

2.运行时间和数据规模并不是线性相关的，原因可能是创建线程开辟内存的时间不稳定。

3.在线程数量较小时，多线程的加速比比单线程要小，但线程数量较大时，加速比比单线程要大。大体而言，多线程的加速比随着线程的增加而增加。

4.多线程的效率整体要比单线程要低。

在进行编程的过程中，我遇到了一些困难，有些是与线程创建相关的知识欠缺，也有一些是程序里不应出现的低级错误。究其原因是自己在并行计算课程知识基础不牢固，以及自身代码能力的欠缺，导致出现在语言上的低级错误。但随着线程数的上升，程序运行时间在一开始时是在不断上升的，这与预期结果相背离，本以为是自己的代码编写错误导致的，于是在起初会强制停止运行，最终造成进度变慢的情况。在与同学和助教的沟通交流后才得以成功完成实验。此后，我也通过修改代码达到缩短运行时间，提高效率的目的。

通过编写三种不同的算法来计算 π 的值，我对计算机资源的调度有了更加深刻的认识。通过对实验串行、并行结果的分析，我也更加清晰地认识到并行化的重要性。这对我未来编写更加高效的代码奠定了更为坚实的基础。

五、课程总结

本次实验是在 Windows 系统和天津大学超算平台的环境下进行实验，而且是一次多

线程编程。虽然有挑战，也遇到了错误，但在多次修改以及查询资料的帮助下，我顺利完成了此次实验。授课上，助教对我给予的帮助很大，实验指导书讲解的也十分详细，对我的实验顺利进行起到了巨大的帮助，使得我可以成功完成本次实验。

不过，在实验过程中，我还是看到了相对不合理的地方。在此请允许我提出一下建议：

1.助教在上机期间偶尔意见难以得到统一，偶尔会出现与实验目的相背离的现象，导致我们的进度相对拖慢。

2.在实验指导书中，建议添加相关知识的网络链接，可以提供一个自主获取知识的渠道，已达到高效完成实验的目的。

3.建议在实验中添加问题探讨，即开放式探究的问题，这样可以促进同学们获取更多的知识。

通过此次实验，我对实验中相关的理论知识和实操技术有了更加深入的了解和巩固。在做实验之前，我认为必须要把课程上学到的知识吃透，因为这是进行实验最重要的基础，否则会对自己实验的难度大大提升，浪费许多宝贵的时间，事倍功半。做实验时，也必须要有不懂的问题虚心请教，可以上网查询或是询问同学或学长。

至此，我顺利完成本次实验的基本任务。实验的过程和思考问题的方法都会对我未来的学习受益匪浅。

附：上机实验与课程知识点分析

序号	上机实验内容	理论知识点	分析总结
1	加速比计算方式	$S(n)=t_s/t_p$	加速比等于串行执行时间与并行执行时间的比值
2	并行计算效率	$E=S(n)/n$	并行计算的效率等于加速比与线程数的比值
3	π的计算公式	三种方式详见实验指导书	积分法和级数法计算的值固定不变，而概率法计算的值不稳定，通过N的上升，可以使π得值趋近于理论值
4			