**高性能并行计算第1次作业**

姓名：姚虎成 学号：2020317110033

**代码地址：**

Hello world的C语言程序路径：

/home2/2020317110033/serial\_1/hello.c

计算pi的C语言程序路径：

/home2/2020317110033/serial\_1/pi.c

**实验结果：**

在计算pi的C程序时，由于计算时间过短，因此在代码实现过程中将计算过程重复了100次。表1是改变不同梯度的steps（104），从1万到1千万取了8个点，用time函数计算了程序运行的时间.图1是将运行时间和Step绘制散点图，并进行了线性拟合。

**表1 Step-Time**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Steps（104） | 1 | 10 | 20 | 50 | 100 | 200 | 500 | 1000 |
| Time（s） | 0.017 | 0.167 | 0.335 | 0.838 | 1.675 | 3.355 | 8.377 | 16.767 |

**图1 Step-Time图**

**实验分析：**

在计算pi的程序中，由于其计算时间过段，即使将积分中的steps提高至10万，时间依然很短，所以在计算程序中将此过程重复一百次来放大时间。

需要将计算PI的C语言程序pi.c编译成可执行程序pi.

编译C语言程序的命令为：gcc -o pi pi.c。

使用Linux下的time命令获取程序运行时间，time命令会得到以下三个时间，分别为real、user、sys时间，real时间是指挂钟时间，也就是命令开始执行到结束的时间。这个短时间包括其他进程所占用的时间片，和进程被阻塞时所花费的时间。sys时间是指花费在内核模式中的CPU时间。user时间是指进程花费在用户模式中的CPU时间，这是唯一真正用于执行进程所花费的时间，其他进程和花费阻塞状态中的时间没有计算在内。我们所需要的是user时间。

最终的图表结果表明，随着Steps的增大，时间也逐渐增大，两者的拟合曲线表明呈线性相关。

**附录：**

计算pi的C语言程序代码

#include<stdio.h>

static long num\_steps=10000;

double step;

void main()

{ int i,j;

for(j=1;j<=100;j++){

double x,pi,sum=0.0;

step=1.0/(double)num\_steps;

for(i=1;i<=num\_steps;i++){

x=(i-0.5)\*step;

sum=sum+4.0/(1.0+x\*x);

}

pi=step\*sum;

}

}