**并行程序设计课程报告**

姓名： 王鹏 学号： 14030130101 日期： 2017.06.08

1. **熟悉MPI并行程序设计环境**
2. 软硬件设置情况

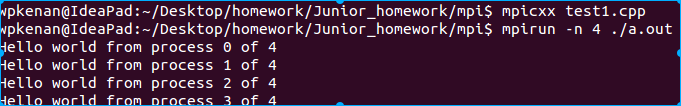
软件版本：32位MPI 1.6.1

共一台多核机器组成模拟并行计算集群

WPKENAN:

1. CPU：Intel® Core™ i5-3337U CPU @ 1.80GHz × 4
2. 内存：4GB
3. IP地址：192.168.0.5
4. 例子程序的运行

Hello world 程序运行



**二、Mandelbrot集的计算**

1. 问题描述

对于非线性迭代公式Zn+1=(Zn)^2+C，所有使得无限迭代后的结果能保持有限数值的复数C的集合，构成曼德勃罗集。Mandelbrot集合就是使以上序列不延伸至无限大的所有c点的集合。从数学上来讲，Mandelbrot集合是一个复数的集合。一个给定的复数c或者属于Mandelbrot集合M，或者不属于。比如，取c = 1，那么这个序列就是(0, 1, 2, 5, 26, ...)，显然它的值会趋于无穷大；而如果取c = i，那么序列就是(0, i, -1+i, -i, -1+i, -i,...)，它的值会一直停留在有限半径的圆盘内。事实上，一个点属于Mandelbrot集合当且仅当它对应的序列（由上面的二项式定义）中的任何元素的模都不大于2。这里的2就是上面提到的“有限半径”。

1. 程序概要设计

Mandelbrot集合是一个无限点集，横坐标范围是 [-2, 2]，纵坐标范围是 [-2, 2]。递推公式是Zn+1=(Zn)^2+C。整个Mandelbrot集各个点的之间的计算并没有先后关系，可以单独计算。所以是一个简单易并行的问题。所以可以利用MPI提供的库例程对这个集合进行计算。

首先假设使用n个进程进行计算。主进程负责收集其他进程计算好的数据，并且负责画图。其余n-1个进程负责计算。为了减少进程之间的通信量，进程处理的最小单位是一行。首先用户的屏幕区域为w\*h。可以设置一个同样大小的二维数组对每个像素点的颜色的进行存储，使用一个缩放函数把像素坐标转换成复数坐标，最后由主进程进行画图。任务分配按行分配。每个从进程处理 h / n个任务。处理的任务范围 [rank \* h / n, rank \* h / n + rank)，如果不能完全分配，也就是n不能整除h，余下部分由主进程全部处理。

总体思路：用户定义的屏幕像素大小（w\*h），然后把每个像素的坐标经过缩放函数映射到复平面坐标。然后使用Mandelbrot集的计算函数计算出该点的迭代次数，取余数可以得到颜色值。最后由主进程进行画图。

多进程任务分配策略：以窗口高度h进行作为分配，假设h = 1000 ，总进程数为4（标号为0,1,2,3）。

P0处理范围 [0, 250)

P1处理范围 [250, 500)

P2处理范围 [500, 750)

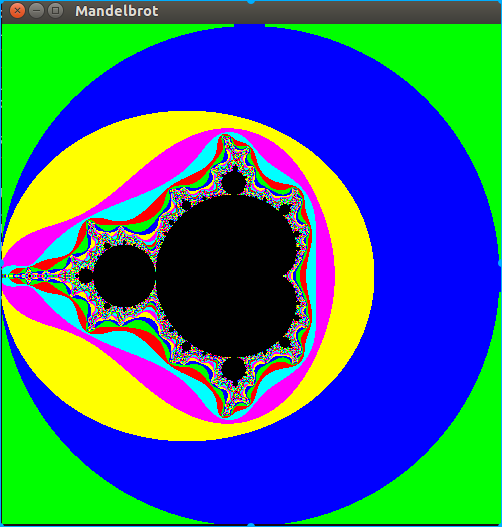
P3处理范围 [750, 1000)

1. 程序运行结果与分析

运行参数说明 Mandelbrot 1000 1000 -2 2 -2 2

窗口大小是1000 \* 1000 复平面坐标范围 实部 [-2, 2] 虚部 [-2, 2]

图像输出



加速比计算

说明：显示图像大小1000 \* 1000

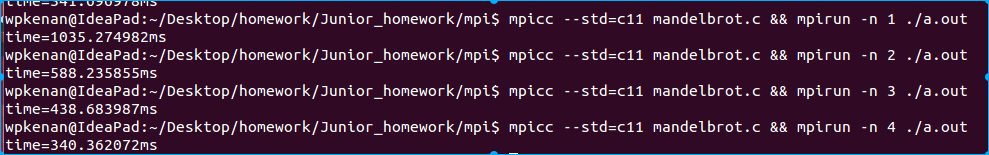
由于一台4核机器，最多同时有4个进程同时运行，所以进程数最高是4

这里的运行时间指的是计算时间。

串行：



并行：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 运行方式 | 串行 | 并行 | | |
| 进程数 | 1 | 1 | 2 | 4 |
| 运行时间（ms） | 1023 | 1035 | 588 | 340 |

分析

当进程数是1 的时候，串行程序和并行程序运行时间并没有差别，由于并行程序需要执行一些mpi的库例程，所以运行时间会稍微长一点。

由Amdahl定律计算加速比

当进程数为2时

sp = 1023 / 588 = 1.74

当进程数为4 时

sp = 1023 / 340 = 3.01

**三、圆周率pi的计算**

1. 问题描述

圆周率（Pi）是圆的周长与直径的比值，一般用希腊字母π表示，是一个在数学及物理学中普遍存在的数学常数。通过积分的方法来计算pi值。积分函数可以选择单位圆 y = sqrt(1 – x^2)，也可以选择arctan的导数 y = 1/(1 + x^2)

1. 程序概要设计

积分的几何意义是曲线与x轴之间围成的面积的大小。对于某个区间的积分，我们可以把区间分成n份，每个进程负责其中一份的计算。计算完各个部分的积分值后，可以使用MPI集合通讯的reduce操作把各进程的计算的面积相加，得到总面积。

求面积的时候可以使用梯形或者长方形近似的方法去求。两种方法的误差不同，留作后面分析。

整体思路：把一个连续的区间离散化为k个小区间，再把k个小区间的面积用梯形或者矩形来近似。可以看到，k值越大，计算精度越高，结果越准确。主要使用的mpi库例程MPI\_Bcast(),MPI\_Reduce()

任务分配策略：每个进程处理区间数 k / n个，余下部分由主进程处理，主进程输出最后结果。

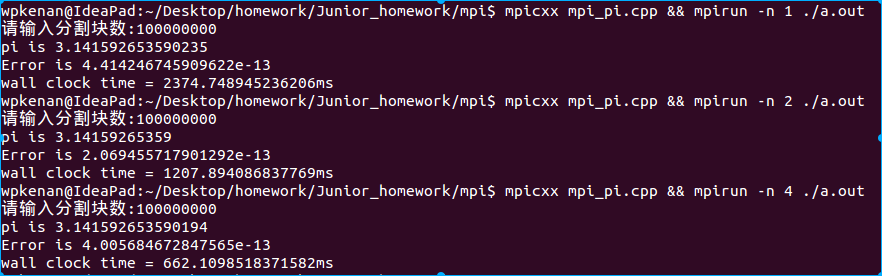
示例：

设共使用4个进程，处理的区间分布如下

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P0 | P1 | P2 | P3 | P0 |

1. 程序运行结果与分析

分割块数 100000000



程序分析：

计算pi值是属于并行度非常高的程序，初始状态，主进程只需要向从进程发送分割总块数，从进程计算完某个区间的面积后，进行相加的归约操作就可以了。所以这个程序的加速比应该能够非常接近理想的加速比。

分割规模 100000000

|  |  |  |  |
| --- | --- | --- | --- |
| 进程数 | 1 | 2 | 4 |
| 运行时间（ms） | 2374 | 1207 | 662 |
| 理想加速比 | 1 | 2 | 4 |
| 实际加速比 | 1 | 1.96 | 3.59 |

实际数据和分析基本一致，所以证明，程序的并行部分越高，使用并行计算效果越好

误差分析：

由于采用了梯形和矩形近似一个小块的面积，必然会产生误差，但是两种方式产生的误差大小不一样。以下是分析。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 分割规模  （误差值10e-16）  近似方式 | 10e1 | 10e3 | 10e5 | 10e7 | 10e9 | 10e11 |
| 梯形 | 16666646826344 | 1666666685 | 166405 | 1941 | 4321 | 848 |
| 矩形 | 8333314113056 | 833333296 | 83684 | 622 | 6333 | 2149 |