**中山大学计算机学院本科生实验报告**

课程名称：超级计算机原理与操作 任课教师：吴迪&黄聃

|  |  |  |  |
| --- | --- | --- | --- |
| 年级 | **2019级** | 专业（方向） | **计算机科学与技术** |
| 学号 | **18329015** | 姓名 | **郝裕玮** |
| 开始日期 | **2021.5.7** | 完成日期 | **2021.5.12** |

1. **实验题目**

nbody问题

1. **实验内容**

实验题目内容为：根据7-Development.pdf课件在nbody问题或者tsp问题中二选一进行实现，要求实现一个串行版本和MPI，OpenMP，pthread中的任意两种版本。

在经过学习和对比之后，我决定实现nbody问题的串行，OpenMP和pthread的三种版本。

**（1）串行版本**

具体分析已全部放在代码注释中，代码如下所示：

#include<iostream>

#include<cmath>   //公式里需要用到sqrt开根函数

#include<fstream> //该头文件用于读写文件

#include<iomanip> //该头文件用于控制数据精度

#include<sys/time.h> //标准日期时间头文件，用于计算运行时间

using namespace std;

#define timestep 20     //迭代次数

#define dT 0.005        //时间间隔

#define G 1             //引力常数

#define MAX 1024        //各信息的数组大小均设置为1024，与粒子个数相同

#define particles 1024   //数据集中共有1024个粒子

#define GET\_TIME(now) { \

   struct timeval t; \

   gettimeofday(&t, NULL); \

   now = t.tv\_sec + t.tv\_usec/1000000.0; \

}

//最后一行将us转换为s，统一单位

//该结构体用于计算程序串行部分运行时间

struct information

{

   double x;

   double y;

   double z;

}f[MAX],pos[MAX],v[MAX];//初始化三个数组，分别为每个粒子的所受作用力，位置，速度

                        //数组内部每个元素存储了对应粒子在x轴，y轴，z轴上的对应分量

int main()

{

   int i,j,k,s;//循环变量

   double x\_diff,y\_diff,z\_diff;//两粒子之间在三条坐标轴上的相对距离

   double dist,dist\_cubed;//两粒子之间的绝对距离和绝对距离的立方

   double x\_f,y\_f,z\_f;//两粒子之间作用力在三条坐标轴上的分量

   double m[MAX];//每个粒子的质量

   double start,end;//用于计算串行部分的开始时间和结束时间

   ifstream myfile("C:\\Users\\93508\\Desktop\\.vscode\\.vscode\\nbody\_init.txt");//打开数据集文件

   for(i=0;i<=particles-1;i++){

      myfile>>m[i]>>pos[i].x>>pos[i].y>>pos[i].z>>v[i].x>>v[i].y>>v[i].z;//根据题目要求将每一行的数据写入到各数组对应的结构体成员中

   }

   GET\_TIME(start);//串行开始

   for(k=1;k<=timestep;k++){//迭代次数为timestep次（20）

      for(i=0;i<=particles-1;i++){//遍历每个粒子

         x\_f=0;

         y\_f=0;

         z\_f=0;

         //每次遍历该粒子以外的粒子时需要先将作用力清空，防止与上一个粒子的作用力相叠加

         for(j=0;j<=particles-1;j++){//遍历该粒子以外的所有粒子

         //因为该粒子以外的所有粒子都对它有作用力

            if(i==j){//该粒子本身可以跳过

               continue;

            }

            x\_diff=pos[i].x-pos[j].x;

            y\_diff=pos[i].y-pos[j].y;

            z\_diff=pos[i].z-pos[j].z;

            //计算两粒子在三条坐标轴上的相对距离

            dist=sqrt(x\_diff\*x\_diff+y\_diff\*y\_diff+z\_diff\*z\_diff);//计算两粒子之间的绝对距离

            dist\_cubed=dist\*dist\*dist;//计算距离的立方

            x\_f-=G\*m[i]\*m[j]/dist\_cubed\*x\_diff;

            y\_f-=G\*m[i]\*m[j]/dist\_cubed\*y\_diff;

            z\_f-=G\*m[i]\*m[j]/dist\_cubed\*z\_diff;

            //计算两粒子之间作用力在三条坐标轴上的分量

         }

         v[i].x+=dT\*x\_f/m[i];

         v[i].y+=dT\*y\_f/m[i];

         v[i].z+=dT\*z\_f/m[i];

         //利用dv=f\*dT/m(动量定理)来更新粒子在三条坐标轴上的速度分量

      }

      for(s=0;s<=particles-1;s++){//遍历每个粒子

         pos[s].x+=dT\*v[s].x;

         pos[s].y+=dT\*v[s].y;

         pos[s].z+=dT\*v[s].z;

         //利用d=d0+v\*dT来更新粒子在三条坐标轴上的坐标

      }

   }

   GET\_TIME(end);//获取串行部分结束时间

   cout<<end-start<<endl;//输出运行时间

   ofstream outfile;

   outfile.open("C:\\Users\\93508\\Desktop\\.vscode\\.vscode\\serial\_result.txt",ios::out);

   //准备将结果写入serial\_result文件中

   for(i=0;i<=particles-1;i++){

      outfile<<setprecision(15)<<m[i]<<' '<<pos[i].x<<' '<<pos[i].y<<' '<<pos[i].z<<' '<<v[i].x<<' '<<v[i].y<<' '<<v[i].z<<endl;

      //将数据保留15位小数并按照要求把每个粒子的数据按顺序写入每行，一行数据代表一个粒子

   }

   myfile.close();

   outfile.close();

   //关闭文件

   system("pause");

}

**（2）OpenMP版本**

OpenMP版本大体内容与串行版本保持一致，唯一的不同是在三个循环前均增加了#pragma omp指令。同时需要注意修改编译参数（eg.在VSCode上编译时，需要在tasks.json中的args里添加一行”-fopenmp”）。

    "args": [

        "-g",

        "${file}",

        "-fopenmp",//需要添加的

        "-o",

        "${fileBasenameNoExtension}.exe"

    ], // 编译命令参数

具体分析已全部放在代码注释中，代码如下所示：

#include<iostream>

#include<cmath>   //公式里需要用到sqrt开根函数

#include<fstream> //该头文件用于读写文件

#include<iomanip> //该头文件用于控制数据精度

#include<sys/time.h> //标准日期时间头文件，用于计算运行时间

using namespace std;

#define timestep 20     //迭代次数

#define dT 0.005        //时间间隔

#define G 1             //引力常数

#define MAX 1024        //各信息的数组大小均设置为1024，与粒子个数相同

#define particles 1024   //数据集中共有1024个粒子

#define GET\_TIME(now) { \

   struct timeval t; \

   gettimeofday(&t, NULL); \

   now = t.tv\_sec + t.tv\_usec/1000000.0; \

}

//最后一行将us转换为s，统一单位

//该结构体用于计算程序串行部分运行时间

struct information

{

   double x;

   double y;

   double z;

}pos[MAX],v[MAX];//初始化两个数组，分别为每个粒子的所受作用力，位置，速度

                 //数组内部每个元素存储了对应粒子在x轴，y轴，z轴上的对应分量

int main()

{

   int i,j,k,s;//循环变量

   double x\_diff,y\_diff,z\_diff;//两粒子之间在三条坐标轴上的相对距离

   double dist,dist\_cubed;//两粒子之间的绝对距离和绝对距离的立方

   double x\_f,y\_f,z\_f;//两粒子之间作用力在三条坐标轴上的分量

   double m[MAX];//每个粒子的质量

   double start,end;//用于计算串行部分的开始时间和结束时间

   int thread\_count;//线程数

   ifstream myfile("C:\\Users\\93508\\Desktop\\.vscode\\.vscode\\nbody\_init.txt");//打开数据集文件

   for(int i=0;i<=particles-1;i++){

      myfile>>m[i]>>pos[i].x>>pos[i].y>>pos[i].z>>v[i].x>>v[i].y>>v[i].z;//根据题目要求将每一行的数据写入到各数组对应的结构体成员中

   }

   thread\_count=8;//在这里选择并行线程数

   GET\_TIME(start);//获取omp并行部分开始时间

# pragma omp parallel num\_threads(thread\_count) \

private(x\_f,y\_f,z\_f,i,j,k,s,x\_diff,y\_diff,z\_diff,dist,dist\_cubed) \

shared(m,pos,v)

//用parallel指令在最外层循环前创建thread\_count个线程的集合。

//同时将上述括号里的变量设置为private（这些变量是每个线程独有的，不可被其他线程访问）

//m,pos,v这三个数组设置成shared，因为粒子的这三个信息会被每个线程访问（并且pos和v的信息在不断更新，所以更加需要被共享）

   for(k=1;k<=timestep;k++){

#        pragma omp for schedule(static)

//使用for指令，告诉OpenMP用已有的线程组来并行化for循环。

//schedule(static)含义为：对for循环并行化进行任务调度

//对于static:这种调度方式非常简单。假设有n次循环迭代，t个线程，那么给每个线程静态分配大约n/t次连续的迭代。

         for(i=0;i<=particles-1;i++){//遍历每个粒子

            x\_f=0;

            y\_f=0;

            z\_f=0;

            //每次遍历该粒子以外的粒子时需要先将作用力清空，防止与上一个粒子的作用力叠加

            for(j=0;j<=particles-1;j++){//遍历该粒子以外的所有粒子

               if(i==j){//该粒子本身可以跳过

                  continue;

               }

               x\_diff=pos[i].x-pos[j].x;

               y\_diff=pos[i].y-pos[j].y;

               z\_diff=pos[i].z-pos[j].z;

               //计算两粒子在三条坐标轴上的相对距离

               dist=sqrt(x\_diff\*x\_diff+y\_diff\*y\_diff+z\_diff\*z\_diff);//计算两粒子之间的绝对距离

               dist\_cubed=dist\*dist\*dist;//计算距离的立方

               x\_f-=G\*m[i]\*m[j]/dist\_cubed\*x\_diff;

               y\_f-=G\*m[i]\*m[j]/dist\_cubed\*y\_diff;

               z\_f-=G\*m[i]\*m[j]/dist\_cubed\*z\_diff;

               //计算两粒子之间作用力在三条坐标轴上的分量

            }

            v[i].x+=dT\*x\_f/m[i];

            v[i].y+=dT\*y\_f/m[i];

            v[i].z+=dT\*z\_f/m[i];

            //利用dv=f\*dT/m(动量定理)来更新粒子在三条坐标轴上的速度分量

         }

#     pragma omp for schedule(static)

      for(s=0;s<=particles-1;s++){//遍历每个粒子

         pos[s].x+=dT\*v[s].x;

         pos[s].y+=dT\*v[s].y;

         pos[s].z+=dT\*v[s].z;

         //利用d=d0+v\*dT来更新粒子在三条坐标轴上的坐标

      }

   }

   GET\_TIME(end);//获取omp并行部分结束时间

   cout<<end-start<<endl;//输出运行时间

   ofstream outfile;

   outfile.open("C:\\Users\\93508\\Desktop\\.vscode\\.vscode\\omp\_result.txt",ios::out);

   //准备将结果写入omp\_result文件中

   for(i=0;i<=particles-1;i++){

      outfile<<setprecision(15)<<m[i]<<' '<<pos[i].x<<' '<<pos[i].y<<' '<<pos[i].z<<' '<<v[i].x<<' '<<v[i].y<<' '<<v[i].z<<endl;

      //将数据保留15位小数并按照要求把每个粒子的数据按顺序写入每行，一行数据代表一个粒子

   }

   myfile.close();

   outfile.close();

   //关闭文件

   system("pause");

}

**（3）pthread版本**

具体分析已全部放在代码注释中，代码如下所示：

1. **实验结果**

代码运行结果及分析