实验三

学号 PB15000027
姓名 韦清

实验题目:
利用 MPI 解决 N 体问题

实验环境(操作系统,编译器,硬件配置等):
操作系统：Ubuntu16.04 （WSL）
编译器：gcc (Ubuntu 5.4.0-6ubuntu1~16.04.9) 5.4.0 20160609
硬件配置：Intel® Core™ i7-8650U CPU @ 1.90GHz

算法设计与分析(写出解题思路和实现步骤)
本次实验算法较为简单，最复杂的地方莫过于计算小球的受力情况了。这里在平面中设置一个 x-y 坐标系，受力情况、速度和位置都为矢量，将它们的 x 值和 y 值分别计算。
计算作用力时，需要将所有其他小球对该小球的力加起来，矢量情况下不好操作，但分为横坐标、纵坐标后就很简单，首先，对每两个小球 i 和 j，计算它们之间的距离 r，通过 r 计算出它们之间的万有引力值 F，F_x = F * distance_x / r，F_y = F * disrance_y / r，F_x 和 F_y 即为这个力在水平方向和竖直方向的值。
此外，计算每个小球受力时，要计算所有其他小球与它的距离情况，但这样会做很多重复计算，如根据小球 j 的位置计算出 j 对 i 的作用力，之后计算 j 受力时，还要重新计算一遍 i 对 j 的作用力，但实际上这两者就算一对相反力，大小相同，方向相反。所以在循环时，对每个 i，只需要计算所有大于 i 的 j，计算出它们受力后，将 i 和 j 所受的这对力都加上了。

```
for(int i=0;i<N;i++) {
    for(int j=i+1;j<N;j++) {
        double distance_x = loc_x[i]-loc_x[j];
        double distance_y = loc_y[i]-loc_y[j];
        double r = sqrt(distance_x*distance_x + distance_y*distance_y);
        double F = calculate_force(r);
        F_x[i] -= distance_x / r * F;
        F_y[i] -= distance_y / r * F;
        F_x[j] += distance_x / r * F;
        F_y[j] += distance_y / r * F;
    }
}
```

计算速度时，使用公式$v = v_0 + at$即可，计算位置时，使用公式 d = vΔt。

在对该算法进行并行化时，主要思想是让每个进程计算 N/size 个小球的受力情况、速度和位置，具体分成三部分：

1. 对 F 的计算
   在对这部分进行并行化时，各进程之间的负载均衡很重要，因为对于每一个小球 i，计算它的力，就要计算所有小球 j(j>i)与它之间的距离，为此，在将 N 分成 size 份时，每个进程都根据它的 id，计算所有 i%size=id 的小球 i。在做计算之前，将所有的小球的受力情况

都初始化为 0，计算完成后，使用 MPI_Reduce 将所有小球的受力加起来，这样效率较高。

2. 对速度的计算
   因为每个小球的速度都只与它自己的受力和初速度有关，和其他小球无关，所以直接跟别计算即可，也不需要将结果输出到其他进程或从其他进程获得输入。
3. 对位置的计算
   位置至于速度有关，所以不需要从其他进程获得输入。但由于计算作用力时需要所有小球的位置，所以计算完成后，还需要将位置发送到主进程，主进程整合后再广播到所有进程。

核心代码(写出算法实现的关键部分,如核心的循环等)

初始化小球位置：

```
void Init(int N) {
    int width = (N==64)?8:16;
    for(int i=0;i<N;i++) {
        v_x[i] = 0;
        v_y[i] = 0;
        F_x[i] = 0;
        F_y[i] = 0;
        //小球初始间隔为 1cm
        loc_x[i] = i/width;
        loc_y[i] = i%width;
    }
}
```

计算作用力：

```
double calculate_force(double r) {
    if(r<RADIUS*2) r = RADIUS*2;
    r = r / 100; //m
    return G*WEIGHT*WEIGHT/(r*r); //N
}

void compute_force(int N, int myid, int size) {
    for(int i=0;i<N;i++) {
        F_x[i] = 0;
        F_y[i] = 0;
    }
    for(int i=myid;i<N;i+=size) {
        for(int j=i+1;j<N;j++) {
            double distance_x = loc_x[i]-loc_x[j];
            double distance_y = loc_y[i]-loc_y[j];
            double r = sqrt(distance_x*distance_x + distance_y*distance_y);//cm
            double F = calculate_force(r);
            F_x[i] -= distance_x / r * F;
            F_y[i] -= distance_y / r * F;
            F_x[j] += distance_x / r * F;
```

```
            F_y[j] += distance_y / r * F;
        }
    }
}
```

计算小球速度：

```
void compute_velocities(int beg, int end) {
    double a_x, a_y;
    for(int i=beg;i<=end;i++) {
        a_x = F_x[i]/WEIGHT; //m*s^-2
        a_y = F_y[i]/WEIGHT; //m*s^-2
        v_x[i] += DELTA * a_x;
        v_y[i] += DELTA * a_y;
    }
}
```

计算小球位置

```
void compute_positions(int beg, int end) {
    for(int i=beg;i<=end;i++) {
        loc_x[i] += DELTA * v_x[i];
        loc_y[i] += DELTA * v_y[i];
    }
}
```

实验结果:

```
root@DESKTOP-UAQ4HIJ:~/MultiProcessing/nbody# mpirun -np 1 ./nbody_mpi
(-6.732113, -6.732113)
time = 6321.279049ms
(-18.910266, -18.910266)
time = 102221.969366ms
root@DESKTOP-UAQ4HIJ:~/MultiProcessing/nbody# mpirun -np 2 ./nbody_mpi
(-6.732113, -6.732113)
time = 1514.425278ms
(-18.910266, -18.910266)
time = 25399.758101ms
root@DESKTOP-UAQ4HIJ:~/MultiProcessing/nbody# mpirun -np 4 ./nbody_mpi
(-6.732113, -6.732113)
time = 2607.985258ms
(-18.910266, -18.910266)
time = 23531.379938ms
root@DESKTOP-UAQ4HIJ:~/MultiProcessing/nbody# mpirun -np 8 ./nbody_mpi
(-6.732113, -6.732113)
time = 3301.010370ms
(-18.910266, -18.910266)
time = 18037.473202ms
root@DESKTOP-UAQ4HIJ:~/MultiProcessing/nbody#
```

对于每个线程数和规模，选取最终结果的一个小球的位置输出，可以看出，在这四种线程下，

结果相同，说明并行结果正确。
下面为运行时间和加速比表格，时间单位为 ms。

运行时间

规模数\线程数	1	2	4	8
64	6321.279049	1514.425278	2607.985258	3301.01037
256	102221.9694	25399.7581	23531.37994	18037.4732

加速比

规模数\线程数	1	2	4	8
64	1	4.174044861	2.423817017	1.914952799
256	1	4.024525311	4.344070328	5.667200069

分析与总结

当小球个数为 64 时，加速比最大为 2 进程并行，4 进程和 8 进程时，由于规模不大，计算开销小，反倒是通信占用了大部分时间，导致加速比不如 2 进程时大。
当小球个数为 256 时，计算开销大，所以 8 进程时加速比最大。

备注(可选)

可选内容如果做了和必选内容一起写在报告中