并行大作业-使用文档及个人报告

张星辰 2000011003

1、我用到的并行技术：

最终版本积分步骤用的是 OpenMP，Scalapack 对角化部分用 MPI.

2.编译安装运行指南：

使用了 APT 安装的 scalapack 和 lapack，用 cmake 编译，相应的 CMakeLists.txt 实现了链接库操作，应该直接链接就可以
程序的结构如下：

```
root@bohrium-12058-1023986:/data/1a_bingxing/aaadazuoye# tree
.
|-- CMakeLists.txt
|-- case1
|   |-- V-1.txt
|   |-- distribution80.txt
|   |-- point1.txt
|   |-- point2.txt
|   `-- point3.txt
|-- include
|   |-- chazhi.h
|   |-- diago.h
|   |-- grid.h
|   |-- input.h
|   |-- scalapack_connector.h
|   `-- timer.h
|-- output.txt
|-- src
|   |-- a.out
|   |-- chazhi.cpp
|   |-- grid.cpp
|   |-- input.cpp
|   |-- main.cpp
|   |-- new.cpp
|   |-- scalapack_diago.cpp
|   |-- scalapack_diago_calculator.cpp
|   `-- timer.cpp
`-- test
    |-- Distribution.txt
    |-- INPUT.txt
    |-- POINTS.txt
    `-- V.txt

4 directories, 26 files
```

每个头文件的用处和结构：

- Input.h:存放用于读入输入文件的输入类
- Timer.h：存放计时类
- Interpolator.h：存放插值类
- Grid.h：存放网格类
- Diago.h：存放对角化用到的 lapack_diago 和 scalapack_diago 函数
- Scalapack_connector.h：存放用到的 scalapack 的函数

关于输入文件：在输入文件进行修改就可以，注意 diago_lib 为 scalapack 时，主程序会讲需要对角化之前的结果输出到 output.txt,需要增加一步对 new.cpp 的编译运行。这里有不止一个 scalapack 的求解器……使用 new.cpp 可能需要手动修改一下 new.cpp 中的矩阵维度和分块矩阵维数。

程序本身的编译说明：使用如下命令构建本项目：

```
# 生成构建目录
cmake -B build
# 执行构建 生成可执行程序
cmake --build build
# 运行程序
./build/myapp
#如果采用 scalapack 进行对角化操作，需要再编译运行一个程序
cd /src
mpicxx new.cpp -lscalapack-openmpi
mpirun --allow-run-as-root -np 4 a.out
```

3.数据结构的设计

程序包含的类有 Input 类、Timer 类、Grid 类、Interpolator 类

1.Input 类用于所有读入工作，该输入类的数据结构存储了程序运行所需的所有参数和数据。下面简要介绍其中一些关键信息：

isHexahedral：标志着输入的网格类型是否为六面体。

lx, ly, lz, thetaxy, thetayz, thetaxz：分别表示网格的长度和三个方向上的倾斜角度。

support_SH：标志着是否支持球谐分析计算。

diago_lib：用于球谐分析计算的库文件名。

support_Periodic_Boundary：标志着是否支持周期性边界条件。

multi_parallel_strategies：标志着使用的并行策略。

points_path：存储了点坐标文件所在的路径。

venergy_path：存储了 V 文件所在的路径。

distribution_path：存储了径向分布函数文件所在的路径。

point1_, point2_：用于指定一个空间子域。

count：点坐标的个数。

nx, ny, nz：网格的 X、Y、Z 方向上的节点数。

cutoff：距离截断半径。

dr：网格步长。

mesh：网格类型。

l：球谐多项式阶数。

f：一维径向分布函数

V：Venergy

在做附加题 1 需要读入多个点时，我把点坐标的形式更改为了结构体，定义一个以结构体为单元的向量用于读入点的坐标

```
struct Point {
double x;
double y;
double z;
};
int count;//点坐标的个数
std::vector<Point> readpointsFile(const std::string& filename, int limit);
```

2.Grid 类的数据结构存储了一个网格的信息，包括了网格的边长及每个方向的节点数等。下面简要介绍其中的变量和函数：

m_lx, m_ly, m_lz：分别表示网格的长度。

m_nx, m_ny, m_nz：分别表示网格的 X、Y、Z 方向上的节点数。

Grid()：构造函数，用于初始化网格对象，输入网格边长 lx, ly, lz 和每个方向点数 nx, ny, nz。

generate_uniform_grid()：生成均匀网格，并返回每个节点的坐标信息。

cell_volume()：计算每个网格单元的体积。

除以上列出的信息外，还有一些其他的函数和变量，用于实现网格操作和提供方便的接口来访问存储在该类中的数据。

3.Interpolator 类：

该插值类的数据结构存储了一组距离和函数值，用于实现插值操作。下面简要介绍其中的变量和函数：

r_：存储了一组距离。

f_：存储了一组函数值。

Interpolator()：构造函数，用于初始化插值对象，输入距离和函数值两个 vector 对象。

distance()：计算点 (x,y,z) 到点 (cx,cy,cz) 的距离。

interpolate()：根据给定距离 r 进行插值操作，返回对应的函数值。

除以上列出的信息外，还有一些其他的函数和变量，用于实现插值操作和提供方便的接口来访问存储在该类中的数据。这个插值类采用的是**三次样条插值方法**。

4.Timer 类：存放计时器，就是本学期之前写的计时器，方便查看效率。

数据结构上的其他考虑：

除了读入都采用向量读入以外，为了避免频繁访问类的成员对象降低效率，我把访问次数>1 的对象（为数组则调用动态数组）存在本地。

4. 程序优化工作（写的比较零散）

这个大作业我写到 6 月第一周快结束的时候中间不小心被我删了，后来重新写的，但是我觉得插值处理的没有原来的好，结果有些奇怪。

依次读入 Input.txt、点坐标、径向分布函数、V 函数。为避免插值积分过程中多次访问 Input 类的成员对象造成效率下降，把需要多次访问的量（如 V、nx、ny、nz、mesh）等存

到本地，插值预处理阶段还需生成一个距离数组 d[mesh]。

调用 Grid 类生成网格（事实上这里网格没有什么实际的意义，原来我还写了一个函数叫 generate_uniform_grid()，后来发现 V1024 总是被 KILLED 就是因为这里把网格点的坐标都给存下来了，会占用很多内存，所以这一步没有了，后续计算插值的时候直接计算网格点的坐标），后调用 cell_volume（）函数返回每个网格单元的体积。

插值阶段和积分阶段重合，定义插值类的对象 interpolator（这里生成插值类需要距离数组 d 和一维径向分布函数数组 f）。插值在计算该格点上积分的时候现行计算，因此可以节约空间成本。这里存在一个五层的嵌套循环，优化时我们知道外面三层应该是动不了的，所以我就只是把内层 r>rcut 的部分取消了计算，节约了一些时间成本。这一部分用 OpenMP 加速。其中 H 矩阵积分点的更新是原子操作。内层的两个循环用#pragma omp simd 向量化了。同时注意了 H 的规约。（注：这一部分是优化的重点内容，发现最后计算量节约下来都是因为这里尽量减少了不必要的运算）

我发现只是注意处理 p=q 的情况，就能使程序从计算 V1024P50 花 1300s 加速到 280s 左右。类似的注意点还有只计算上三角部分。

计算完上三角部分后我们把下三角部分对称的赋值。之后所有格点积分乘以 cell_vol，这样可以减少在关键部分作乘法的数量。之后进行对角化操作得到最终结果。

```
#pragma omp parallel for reduction(+ : H[ : count * count])
    for (int i = 0; i < nx; i++)
    {
        for (int j = 0; j < ny; j++)
        {
            for (int k = 0; k < nz; k++)
            {

                double x0 = i * dx;

                double y0 = j * dy;

                double z0 = k * dz;

                double v = V[(i * ny + j) * nz + k];//不过反正V都是1
                //double v = 1.0;
#pragma omp simd
                for (int p = 0; p < count; p++)
                {
                    // 现算p和q点的插值
                    double r1 = interpolator.distance(x0, y0, z0, points_[p].x, points_[p].y, points_[p].z);

                    if (r1 < cutoff) // 如果f1还是0就直接不算了
                    {
                        double f1 = interpolator.interpolate(r1);
#pragma omp simd
                        for (int q = 0; q <= p; q++)
                        {
                            if (q == p)
                            {
#pragma omp atomic
                                H[p * count + q] += f1 * f1 * v;
                                continue;
                            }
                            double r2 = interpolator.distance(x0, y0, z0, points_[q].x, points_[q].y, points_[q].z);

                            if (r2 < cutoff)
                            {
                                double f2 = interpolator.interpolate(r2);
#pragma omp atomic
                                H[p * count + q] += f1 * f2 * v;
                            }
                        }
                    }
                }
            }
        }
    }
```

代码关键部分我是用的 OpenMP，但是把 scalapack 整合后影响了原来的效率。我整合了 MPI 进去的 V1024P50，比原来慢了好多，推测是由于线程冲突，越并行越慢……

所以我后来尝试**把 scalapack 的功能单独写了一个文件**，避免线程冲突影响原来的效率，一开始放在了 scalapack_diago_caculator.cpp 里，但是这个程序我用到了一个函数，声明如下：
void pdgemr2d_(int *m, int *n, double *a, int *ia, int *ja, int *desca,double *b, int *ib, int *jb, int *descb,int *context, int *info);
这个函数我在使用的时候后来一直报错，查看了手册等很多资料也没有解决。后来重新换用了 new.cpp 作为求解器，但是时间有限，这个求解器也有点瑕疵，需要手动修改矩阵的维度和分块矩阵维度。

4.测试报告

正确性：

按照正确性测试的结果来看是正确的。

效率：（8 核）

V512 Points2 运行约 2s

```
|CLASS_NAME---------|NAME---------------|TIME(Sec)-----|CALLS----|AVG--------|PER%-------
                                total        3.06589         1   3.06589           100%
               INPUT                Input       0.001183         1  0.001183      0.0385859%
               INPUT            parseFile       0.001174         1  0.001174      0.0382924%
               INPUT       readpointsFile       0.001965         1  0.001965      0.0640924%
               INPUT     readDirstibution       0.003501         1  0.003501       0.114192%
                Grid          cell_volume          1e-06         1     1e-06     3.2617e-05%
                 aaa                jifen        2.10171         1   2.10171        68.5513%
        Interpolator          interpolate        1.02041    4138442.46568e-06        33.2826%
-------------------------------------------------------------------------------------------
```

V512 Points50 运行约 37s

```
|CLASS_NAME---------|NAME---------------|TIME(Sec)-----|CALLS----|AVG--------|PER%-------
                                total         38.622         1    38.622           100%
               INPUT                Input       0.001327         1  0.001327     0.00343587%
               INPUT            parseFile       0.001318         1  0.001318     0.00341256%
               INPUT       readpointsFile        0.00161         1   0.00161     0.00416861%
               INPUT     readDirstibution       0.002668         1  0.002668     0.00690798%
                Grid          cell_volume          1e-06         1     1e-06     2.5892e-06%
                 aaa                jifen        37.6649         1   37.6649        97.5219%
        Interpolator          interpolate        19.5934  102202591.91711e-06        50.7312%
-------------------------------------------------------------------------------------------
```

V1024 Points2 运行约 17s

```
|CLASS_NAME---------|NAME---------------|TIME(Sec)-----|CALLS----|AVG--------|PER%-------
                                total        24.9517         1   24.9517           100%
               INPUT                Input       0.000992         1  0.000992     0.00397568%
               INPUT            parseFile       0.000983         1  0.000983     0.00393961%
               INPUT       readpointsFile       0.000909         1  0.000909     0.00364304%
               INPUT     readDirstibution       0.002266         1  0.002266     0.00908155%
                Grid          cell_volume          1e-06         1     1e-06    4.00775e-06%
                 aaa                jifen        17.3453         1   17.3453        69.5154%
        Interpolator          interpolate        8.36979   33212582.52007e-06         33.544%
-------------------------------------------------------------------------------------------
```

V1024 Points10 运行约 142s

```
|CLASS_NAME---------|NAME---------------|TIME(Sec)-----|CALLS----|AVG--------|PER%-------
                                total       150.051         1   150.051           100%
               INPUT              Input       0.000988         1  0.000988    0.000658444%
               INPUT          parseFile       0.000978         1  0.000978     0.00065178%
               INPUT      readpointsFile       0.018891         1  0.018891      0.0125897%
               INPUT    readDirstibution       0.002824         1  0.002824     0.00188203%
                Grid         cell_volume          2e-06         1     2e-06    1.33288e-06%
                 aaa               jifen       142.455         1   142.455        94.9377%
         Interpolator         interpolate       73.1624  387606681.88754e-06        48.7584%
-------------------------------------------------------------------------------------------
```

V1024,Points50 运行约 283s

```
|CLASS_NAME---------|NAME---------------|TIME(Sec)-----|CALLS----|AVG--------|PER%-------
                                total       291.067         1   291.067           100%
               INPUT              Input       0.002136         1  0.002136    0.000733852%
               INPUT          parseFile       0.002127         1  0.002127     0.00073076%
               INPUT      readpointsFile       0.003258         1  0.003258     0.00111933%
               INPUT    readDirstibution       0.002328         1  0.002328    0.000799816%
                Grid         cell_volume          1e-06         1     1e-06    3.43564e-07%
                 aaa               jifen       283.481         1   283.481        97.3939%
         Interpolator         interpolate       148.751  820443541.81306e-06        51.1056%
-------------------------------------------------------------------------------------------
```