README

工学院 2100011029 渠成锐

2023 年 6 月 24 日

1 并行技术

本版本为 cuda 版本。

2 编译运行指南

2.1 编译运行指令

本项目采用 Makefile 进行构建，进入项目目录 (cuda_version) 后在命令行运行如下指令：

1. make cuda
2. ./cuda

main 函数将生成 matrix.txt 文件记录积分得到的实对称矩阵及维数，存放至 output 文件夹中。

对于对角化，只需将生成的 matrix.txt 文件 (连并 output 文件夹一起) 与 scalapack.cpp 文件移到前文 openmp 编译过程中配好的环境里，运行如下指令：

1. mpicxx scalapack.cpp -lscalapack-openmpi
2. mpirun -np 4 ./a.out

该程序将读入 matrix.txt 并利用 scalapack 接口进行对角化, 将特征值和特征向量输出至 output 文件夹中。(注：本程序默认用 4 核进行对角化)

3 文件结构

1. input(文件夹)：存放输入文件
2. output(文件夹)：存放输出文件
3. tools(文件夹)：存放用到的库

3.1 代码结构

1. main.cu: 主函数
2. input.h: input 类，实现读入文件输入
3. spline.h: spline 类，实现三次样条插值
4. scalapack.cpp: 调用 scalapack 接口，采用并行方式对实对称矩阵对角化

3.2 输入文件结构

1. INPUT.txt: 由 input 类读取的输入文件，设置计算任务及输入矩阵文件路径
2. 调试时将空间相关输入文件放置在 input 文件夹中，并将 INPUT.txt 中的路径设置为相应路径即可。

3.3 输出文件结构

输出文件均存放至 output 文件夹中

1. matrix.txt：积分得到的矩阵，第一行为维数，后面为矩阵元

4 数据结构的设计

设备端静态声明的数组如下：

```
1    __device__ double fvalues[2000];//存放势函数分布
2    __device__ double m[2000];//存放插值节点导数值
3    __device__ int dev_point[2500]; //存放要计算的点对
```

设备端动态声明的数组包括：

```
int* dev_area;//存放点对重合的空间区域起始点坐标
float* dev_matH;//hamilton矩阵
double* dev_V;//V数组
```

注：最占内存的是 V 数组，其他数组的存储空间相比之下都小的多；此外需要说明的是，这里 Hamilton 矩阵采用 float 型存储是因为 device 端的 atomicAdd 原子操作只能对 float 型操作，并不支持 double 类型。

5 优化工作

首先，为了减小计算量，main 函数中首先进行了预处理，对于所有点对，计算出点对是否相交（即两点距离是否大于二倍的 cutoff），以及需要计算的点对重叠的区域（为了简便计算，我们视以每个点为体心张开一个边长为 2 倍 cutoff 的正方体，通过判断正方体是否相交来得到点对是否相交以及相交的区域），由此可以大大减少需要积分计算的区域。

此外，为了各个 block 内线程负载均衡，我们试图让每个点对重叠的区域为大小相同（即可以分的粗一点），我们首先利用 cutoff 的大小和空间小格点的边长大小 (lx/nx) 得到边长为 2 倍 cutoff 的正方体边长上的点的数量 (width, 设计为偶数，便于分割任务)，然后我们每一个 block 计算一个点对的重叠区域，重叠区域大小为一个正方体，边长为 width(int 型，表示点的数量)，并设计一个 block 内起（width*4）个线程，这样每个线程需要计算的空间格点的数量就是相同的 width*width/4。(我们假设测试用的 GPU 一个 block 内最多能起的线程数为 1024，在如上设计下，对于空间分的最细的情况下，width 约为 166，确保了起的线程数不会超过 1024)。

本项目对 50 个点，V-512 的情况进行了测试，积分步用时 60ms 左右。(如果助教测试时遇到任何问题可以直接联系我，我应该都在学校)