并行计算实验报告 – 1

运行命令

python Random_matrix.py 10 10

./Reduction 10 10 –t –p –d

mpiexec –n 5 ./Reduction_mpi 10 10 –t –p –d

./Validation_mpi -p

原算法思路

- 原算法接收最多 5 个参数：前两个为 M 和 N 表示矩阵的行数和列数，然后是-t，打印运行时间；-p 打印结果矩阵；-d 将结果矩阵分别存入文件。
- 原算法第 43 到 52 行为 U_t、Alphas、Betas、Gammas 各申请了 N*N 的空间，但 65 到 70 行的 for 循环中将数据读入到 M*N 的矩阵中，而后面 92 到 94 行又将计算结果存入矩阵，访问的范围是 M*M（计算过程也是从 M*M 的矩阵中获取数据）。因此，当 M>N 时原始算法不能正常运行，所以进行实验时假设原始算法正确并且保证 M<N，在程序中进行检测，并在不满足时打印错误信息。
- 核心算法代码是 78 到 97 行的三层循环结构，且核心语句使用了全部三个循环变量 i、j、k。

并行改造思路

核心思想就是将原始算法第 78 行的循环拆解，将对 M 行的计算工作平均分配给其他进程，每一进程需要计算的行数=(M + size – 1) / size。

在计算之前，主进程（MASTER 进程，rank=0）利用 MPI Broadcast 将输入矩阵广播给其他所有进程，为了方便进程间数据传输，将所有矩阵声明为一维数组，即将点 U_t（i，j）映射到 U_t‘（i * N + j）。另外，主进程开始计算任务之前先要使

用异步接收函数 MPI_Irecv 设置接收监听，以便其他进程计算完成之后能及时将数据回传给主进程汇总。

计算时，每个进程只计算 rank * rowPerRank 到(rank + 1) * rowPerRank 之间的行的数据。

计算之后将自己计算过的行的数据回传给主进程。同时主进程要等待所有进程的数据都回传之后才会进行下一步操作（结束计时，输出结果等）。

运行结果

```
@@@ 0 @@@ Broadcasted!
@@@ 0 @@@ rowPerRank = 2
@@@ 0 @@@ before set Irecv
@@@ 0 @@@ setting Irecv for 1
@@@ 0 @@@ Irecv for 1 set!
@@@ 0 @@@ setting Irecv for 2
@@@ 0 @@@ Irecv for 2 set!
@@@ 0 @@@ setting Irecv for 3
@@@ 2 @@@ Broadcasted!
@@@ 2 @@@ rowPerRank = 2
@@@ 2 @@@ Start computing...
@@@ 2 @@@ End computing!
@@@ 2 @@@ before send data
@@@ 2 @@@ _size = 20
@@@ 2 @@@ sending data
@@@ 4 @@@ Broadcasted!
@@@ 4 @@@ rowPerRank = 2
@@@ 4 @@@ Start computing...
@@@ 4 @@@ End computing!
@@@ 4 @@@ before send data
@@@ 4 @@@ _size = 20
@@@ 4 @@@ sending data
@@@ 1 @@@ Broadcasted!
@@@ 1 @@@ rowPerRank = 2
@@@ 1 @@@ Start computing...
@@@ 1 @@@ End computing!
@@@ 1 @@@ before send data
@@@ 1 @@@ _size = 20
@@@ 1 @@@ sending data
@@@ 3 @@@ Broadcasted!
@@@ 3 @@@ rowPerRank = 2
@@@ 3 @@@ Start computing...
@@@ 3 @@@ End computing!
@@@ 3 @@@ before send data
@@@ 3 @@@ _size = 20
@@@ 3 @@@ sending data
@@@ 1 @@@ exit!
@@@ 2 @@@ exit!
@@@ 4 @@@ exit!
@@@ 3 @@@ exit!
@@@ 0 @@@ Irecv for 3 set!
@@@ 0 @@@ setting Irecv for 4
@@@ 0 @@@ Irecv for 4 set!
@@@ 0 @@@ set Irecv!
@@@ 0 @@@ Start computing...
@@@ 0 @@@ End computing!
Time: 0.11 ms.

FQs-MacBook:1 fanquan$
```

```
FQs-MacBook:1 fanquan$ ./Validation_mpi -p
VALID!

difference in Alphas: 0
difference in Betas: 0
difference in Gammas: 0
```