计算物理 A 第十八题

杨旭鹏 PB17000234

2019 年秋季

1 题目描述

设计一个 MPI 程序，满足：

1. 各进程随机生成一个 10×10 二维单精度浮点数组，且各进程中的数组对应元素值的不全一样
2. 调用集合通信 MPI_Reduce 函数实现对数组中对应元素求最大值的归约操作
3. 利用点对点通讯实现上述归约操作并与其进行验证

编译、测试运行可使用 https://training.ustc.edu.cn/ 中的《MPI 编译环境的使用》

2 程序思路

首先初始化并行环境 MPI_Init 。在主进程中产生供不同进程使用的不同随机种子，存在一个数组中，利用集合通信（MPI_Bcast）将此数组共享给其他进程。然后利用循环对所有位置的矩阵元求最大值，利用两种方法求某位置矩阵元的最大值：

1. 利用集合通信：将每一进程的某矩阵元传入 MPI_Reduce 函数，并发送到最后一个进程中，设定 MPI_Op 操作为为 MPI_MAX，然后在最后一个进程中打印传入的寻找到的最大值矩阵即为所求。
2. 利用点对点通信：将每一非最后一个进程中的某矩阵元数值传入最后一个进程中，在最后一个进程接收到一数组中，然后比较数组中的元素，将最大值存入某数组，打印，即为所求。

3 程序说明

该程序中矩阵元均为 $(0,1)$ 的随机单精度浮点数，且初始种子与时间有关，故此程序每次运行结果均不同。按照老师给的示例，需要将所有的矩阵打印出来，为了保证进程间打印不互相干扰，故程序设有打印的标志 flag，标志值不为 0 时才执行打印操作。初始只有主进程 flag 不为 0，其余进程打印之前接受上一进程传来的 flag 置 1 信号。打印完成之后利用点对点通信将下一个进程的 flag 置 1，以完成顺序打印随机矩阵。

编译命令为：mpicc −o mpi mpi.c，运行命令按照李老师给的示例为：mpirun -n 4 ./mpi ，默认开启 4 个进程。

4 程序结果

将本程序源代码文件“mpi.c”上传到中国科大计算实训平台上，然后编译运行。得到如下结果：

```
[p-22017040010a2e845d515f 2201704001]$ mpicc -o mpi mpi.c
[p-22017040010a2e845d515f 2201704001]$ mpirun -n 4 ./mpi
进程号:0的随机矩阵为：
0.189   0.016   0.483   0.135   0.856   0.881   0.318   0.736   0.888   0.633
0.528   0.668   0.149   0.521   0.146   0.066   0.851   0.080   0.230   0.904
0.837   0.256   0.238   0.425   0.334   0.518   0.657   0.856   0.916   0.051
0.873   0.105   0.067   0.356   0.239   0.923   0.237   0.558   0.660   0.125
0.190   0.188   0.793   0.339   0.709   0.938   0.405   0.559   0.019   0.635
0.464   0.856   0.892   0.701   0.282   0.226   0.219   0.938   0.082   0.135
0.990   0.955   0.240   0.057   0.311   0.479   0.980   0.548   0.036   0.640
0.673   0.227   0.827   0.465   0.566   0.536   0.404   0.971   0.095   0.422
0.606   0.559   0.279   0.498   0.260   0.560   0.724   0.480   0.498   0.806
0.614   0.488   0.761   0.854   0.545   0.072   0.333   0.525   0.620   0.369
进程号:1的随机矩阵为：
0.237   0.288   0.163   0.581   0.415   0.103   0.872   0.875   0.087   0.867
0.282   0.390   0.404   0.577   0.863   0.663   0.166   0.673   0.456   0.578
0.671   0.144   0.684   0.849   0.644   0.739   0.380   0.752   0.649   0.808
0.660   0.886   0.096   0.823   0.467   0.511   0.926   0.339   0.386   0.013
0.206   0.668   0.403   0.610   0.245   0.267   0.274   0.410   0.939   0.729
0.988   0.610   0.873   0.672   0.459   0.517   0.411   0.839   0.269   0.060
0.647   0.929   0.946   0.743   0.752   0.412   0.254   0.678   0.751   0.640
0.692   0.957   0.308   0.095   0.567   0.552   0.362   0.841   0.963   0.302
0.570   0.951   0.912   0.443   0.624   0.371   0.961   0.035   0.210   0.230
0.095   0.857   0.159   0.041   0.600   0.911   0.453   0.854   0.589   0.204
进程号:2的随机矩阵为：
0.378   0.643   0.511   0.915   0.370   0.499   0.168   0.192   0.761   0.761
0.583   0.001   0.473   0.054   0.257   0.807   0.245   0.180   0.231   0.253
0.462   0.311   0.828   0.360   0.033   1.000   0.995   0.682   0.380   0.472
1.000   0.758   0.116   0.510   0.673   0.486   0.010   0.842   0.677   0.771
0.603   0.261   0.772   0.076   0.315   0.029   0.882   0.560   0.208   0.114
0.813   0.671   0.425   0.641   0.031   0.458   0.641   0.026   0.140   0.021
0.498   0.140   0.779   0.614   0.650   0.452   0.100   0.660   0.294   0.777
0.431   0.897   0.038   0.203   0.973   0.353   0.231   0.855   0.912   0.440
0.969   0.725   0.110   0.394   0.366   0.141   0.852   0.007   0.168   0.992
0.028   0.666   0.132   0.807   0.280   0.782   0.259   0.380   0.442   0.553
进程号:3的随机矩阵为：
0.293   0.508   0.976   0.224   0.433   0.843   0.369   0.338   0.280   0.239
0.895   0.329   0.203   0.669   0.419   0.795   0.344   0.306   0.380   0.436
0.810   0.223   0.713   0.980   0.131   0.982   0.525   0.420   0.259   0.998
0.920   0.552   0.506   0.897   0.776   0.940   0.740   0.145   0.277   0.020
0.384   0.172   0.350   0.587   0.841   0.769   0.382   0.185   0.075   0.761
0.622   0.885   0.984   0.334   0.865   0.115   0.317   0.390   0.535   0.576
0.388   0.455   0.128   0.894   0.352   0.904   0.834   0.092   0.049   0.111
0.112   0.433   0.283   0.462   0.020   0.124   0.230   0.402   0.309   0.305
0.163   0.931   0.190   0.147   0.266   0.055   0.263   0.582   0.444   0.798
0.158   0.832   0.253   0.287   0.727   0.604   0.191   0.560   0.696   0.240
集合通信：
0.378   0.643   0.976   0.915   0.856   0.881   0.872   0.875   0.888   0.867
0.895   0.668   0.473   0.669   0.863   0.807   0.851   0.673   0.456   0.904
0.837   0.311   0.828   0.980   0.644   1.000   0.995   0.856   0.916   0.998
1.000   0.886   0.506   0.897   0.776   0.940   0.926   0.842   0.677   0.771
0.603   0.668   0.793   0.610   0.841   0.938   0.882   0.560   0.939   0.761
0.988   0.885   0.984   0.701   0.865   0.517   0.641   0.938   0.535   0.576
0.990   0.955   0.946   0.894   0.752   0.904   0.980   0.678   0.751   0.777
0.692   0.957   0.827   0.465   0.973   0.552   0.404   0.971   0.963   0.440
0.969   0.951   0.912   0.498   0.624   0.560   0.961   0.582   0.498   0.992
0.614   0.857   0.761   0.854   0.727   0.911   0.453   0.854   0.696   0.553
点对点通信：
0.378   0.643   0.976   0.915   0.856   0.881   0.872   0.875   0.888   0.867
0.895   0.668   0.473   0.669   0.863   0.807   0.851   0.673   0.456   0.904
0.837   0.311   0.828   0.980   0.644   1.000   0.995   0.856   0.916   0.998
1.000   0.886   0.506   0.897   0.776   0.940   0.926   0.842   0.677   0.771
0.603   0.668   0.793   0.610   0.841   0.938   0.882   0.560   0.939   0.761
0.988   0.885   0.984   0.701   0.865   0.517   0.641   0.938   0.535   0.576
0.990   0.955   0.946   0.894   0.752   0.904   0.980   0.678   0.751   0.777
0.692   0.957   0.827   0.465   0.973   0.552   0.404   0.971   0.963   0.440
0.969   0.951   0.912   0.498   0.624   0.560   0.961   0.582   0.498   0.992
0.614   0.857   0.761   0.854   0.727   0.911   0.453   0.854   0.696   0.553
```

图 1: 程序运行过程示意

其中，最后输出的 2 个矩阵分别为两种方式求得的最大值矩阵。可以看出两个方法均能求出最大值矩阵。程序成功。

5 心得体会

虽然本次作业思路上相比之前的作业简单不少，但是毕竟是对新的语法进行编程，花费的时间也不算太少。编程中发现 MPI_Reduce 函数应在每个进程中都运行，一开始只在一个进程中运行发现结果上有问题。还有就是程序间打印顺序的问题也调试了很久。总之确实是熟悉了一下 MPI 的基本语法。

6 C 语言程序源代码

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include "mpi.h"
#define GETMATELEM(base,i,j,imax) ((*(base + (i) * (imax) + (j)))) //取二维数组元素


int matrixform(float *matrix,int m,int n){ //打印矩阵的函数
    int i,j;
    for(i=0;i<m;i++){
        for(j=0;j<n;j++){
            printf("%.3f\t", GETMATELEM(matrix,i,j,n) );
        }
        printf("\n");
    }
    return 0;
}



int main(int argc, char* argv[]){

    int m; //求最大值的矩阵元行位置
    int n; //求最大值的矩阵元列位置
    float *max1 = malloc(sizeof(float)*100); //利用集合通信求得的最大值矩阵
    float *max2 = malloc(sizeof(float)*100); //利用点对点通信求得的最大值矩阵
    int flag = 0; //打印矩阵的标志
    int one = 1; //值为 (int) 1 的变量

    int numprocs, myid, source;
```

```
32    MPI_Status status;
33    MPI_Init(&argc,&argv);
34    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
35    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
36
37    float *max2temp = malloc(sizeof(float)*numprocs);
          //点对点通信时比较大小所用的临时数组
38    unsigned int seed[numprocs]; //随机种子数组
39
40    if(myid == 0){
41
42        time_t t;
43        srand((unsigned) time(&t));
44        int i;
45        for(i=0;i < numprocs;i++){ //种子初始化
46            seed[i] = rand();
47        }
48        srand(seed[myid]);
49        flag = 1;
50
51    }
52
53    MPI_Bcast(seed, numprocs, MPI_UNSIGNED, 0, MPI_COMM_WORLD);
          //广播主进程求得的随机种子数组
54
55    srand(seed[myid]);      //对不同进程赋不同的随机种子
56    float *matrix = malloc(sizeof(float)*100); //为每一个进程定义随机矩阵
57    int j,k;
58    for(j=0;j<10;j++){   //对随机矩阵赋值
59        for(k=0;k<10;k++){
60            GETMATELEM(matrix,j,k,10) = rand()/(float)RAND_MAX;
                  //矩阵元为 (0, 1) 随机f单精度浮点数
61        }
62    }
63
64
65    if(myid != 0){
66        MPI_Recv(&flag, 1, MPI_INT, myid -1 , 1, MPI_COMM_WORLD,
              &status);//接受上一个进程传入的打印矩阵flag参数
67    }
68
69
70    if(flag != 0){ //若打印flag不为0, 则执行打印矩阵命令
71        printf("进程号:%d的随机矩阵为:\n",myid);
72        matrixform(matrix,10,10);
```

```
73          if(myid < numprocs -1){
74              MPI_Send(&one, 1, MPI_INT, myid + 1, 1, MPI_COMM_WORLD);
                    //向下一个进程传入的打印矩阵flag参数
75          }
76      }
77
78
79
80      for(m=0;m<10;m++){ //对矩阵行元素循环
81            for(n=0;n<10;n++){ //对矩阵列元素循环
82                MPI_Reduce(&GETMATELEM(matrix,m,n,10), &GETMATELEM(max1,m,n,10), 1,
                        MPI_FLOAT, MPI_MAX, numprocs-1, MPI_COMM_WORLD);
                        //集合通信求最大值矩阵
83
84                if(myid != numprocs -1){
85                    MPI_Send(&GETMATELEM(matrix,m,n,10), 1, MPI_FLOAT, numprocs -1,
                            myid+10, MPI_COMM_WORLD); // 点对点通信
86                }
87
88                if(myid == numprocs-1){ //最后一个进程中利用点对点通信求最大值
89                    int i;
90                    GETMATELEM(max2,m,n,10) = GETMATELEM(matrix,m,n,10);
91                    for(i = 0;i<numprocs-1;i++){ //点对点接受然后求最大值
92                        MPI_Recv(&max2temp[i], 1, MPI_FLOAT, i, i+10, MPI_COMM_WORLD,
                                &status);
93                        if(max2temp[i] > GETMATELEM(max2,m,n,10) ){
94                            GETMATELEM(max2,m,n,10) = max2temp[i];
95                        }
96                     }
97
98                }
99            }
100       }
101
102
103     if(myid == numprocs-1){ //最后一个进程打印最大值矩阵
104         printf("集合通信:\n");
105         matrixform(max1,10,10);
106         printf("点对点通信:\n");
107         matrixform(max2,10,10);
108     }
109
110     MPI_Finalize();
111 }
```