HW18

古宜民

2020.1.2

题目

设计一个MPI程序：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不全一样，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。结果是一个10×10数组，比如其(1,3)元素存储的是各进程自己的10×10数组中对应(1,3)元素最大的，不是各进程求自己的单个数组中最大的。利用点对点通讯实现上述归约操作并与其进行验证。

环境搭建

在Archlinux系统的笔记本上，直接安装编译好的openmpi环境：

```
sudo pacman -S openmpi
```

检查环境：

```
➜  ~ mpicc --showme:version
mpicc: Open MPI 4.0.2 (Language: C)
```

程序设计

由题目可知，如果使用N个进程，则其中有N-1个进程生成随机矩阵，1个进程（认为是父进程）用于统计。这N-1个进程将自己的每一个元素(i,j)使用`MPI_Send`发送给父进程，父进程对于矩阵的第(i,j)元素，接收来自N-1个进程的N-1个信号，并计算这N-1个信号中传来的浮点数的最大值。

- N-1个子进程的主要代码：

```
        srand((unsigned)time(0) * rank);
        float randmat[N][N];
        int i, j;
        char filename[100];
        sprintf(filename, "data_%d.txt", rank);
        FILE* fout = fopen(filename, "w");
        fprintf(fout, "Child %d\n", rank);
        for (i = 0; i < N; i++) {
            for(j = 0; j < N; j++) {
                randmat[i][j] = rand() / (double) RAND_MAX;
                fprintf(fout, "%f ", randmat[i][j]);
            }
            fprintf(fout, "\n");
        }
        fclose(fout);
        for (i = 0; i < N; i++) {
            for(j = 0; j < N; j++) {
```

```
                MPI_Send(&randmat[i][j], 1, MPI_FLOAT, 0, i * N + j,
MPI_COMM_WORLD);
            }
        }
```

首先生成随机数。为了避免不同进程的种子相同，这里的种子使用了进程编号。

第一个循环生成随机矩阵，并为了查看结果将其输出到文件。

第二个循环发送消息。其中为了保证同一个子进程不同元素对应的tag的唯一，用了i*N+j作为tag。

- 父进程消息处理的代码：

```
        float resultmat[N][N];
        int i, j, k;
        printf("Father start...\n");
        int childid;
        for (i = 0; i < N; i++) {
            for(j = 0; j < N; j++) {
                float max = -9e9;
                float buf;
                MPI_Status status;
                for (k = 1; k < size; k++) {
                    /*printf("Father>Prepare to receive signal\n");*/
                    MPI_Recv(&buf, 1, MPI_FLOAT, k, i * N + j, MPI_COMM_WORLD,
&status);
                    /*printf("Father>Signal received.\n");*/
                    max = buf > max ? buf : max;
                }
                resultmat[i][j] = max;
            }
        }
        printf("Maximum: \n");
        for (i = 0; i < N; i++) {
            for(j = 0; j < N; j++) {
                printf("%f ", resultmat[i][j]);
            }
            printf("\n");
        }
        printf("Father end. \n");
```

第一组循环对每个元素接受rank-1（即N-1）个消息，然后求得最大值。

第二个循环输出。

运行结果

编译

```
mpicc hw18.c
```

运行，用4个进程

```
mpirun -np 4 ./a.out
```

输出结果：

```
Father start...
Maximum:
0.902092 0.343041 0.998444 0.698059 0.909447 0.931358 0.551412 0.136895 0.699539
0.543107
0.831910 0.968319 0.976954 0.298890 0.679858 0.780468 0.865718 0.762059 0.949442
0.733858
0.231281 0.911227 0.738972 0.991756 0.743202 0.841439 0.903767 0.646212 0.612873
0.994891
0.848370 0.556774 0.352937 0.698128 0.784767 0.687230 0.800783 0.865992 0.768730
0.934548
0.707875 0.838086 0.804938 0.648476 0.863757 0.484796 0.915685 0.729475 0.704010
0.732730
0.796557 0.776297 0.536330 0.660176 0.457447 0.844923 0.799376 0.537157 0.939108
0.602723
0.667453 0.590004 0.975411 0.986941 0.872369 0.423799 0.970678 0.715987 0.920530
0.791620
0.607699 0.955491 0.360605 0.889654 0.627929 0.591019 0.940091 0.755738 0.818823
0.601443
0.851274 0.959908 0.978393 0.269688 0.250596 0.835187 0.940356 0.887210 0.945093
0.879465
0.985446 0.944275 0.936515 0.960857 0.599487 0.818846 0.852224 0.366134 0.577105
0.772754
Father end.
```

各个子矩阵的文件：

```
➜  HW18 git:(master) ✗ cat data*.txt
Child 1
0.046505 0.320434 0.849758 0.592539 0.371904 0.102655 0.551412 0.049846 0.133765
0.334236
0.831910 0.870391 0.613515 0.172868 0.679858 0.193054 0.532278 0.219214 0.564248
0.733858
0.072287 0.911227 0.652186 0.681150 0.743202 0.343274 0.599651 0.265663 0.510268
0.994891
0.848370 0.556774 0.315325 0.698128 0.149313 0.687230 0.800783 0.700725 0.737076
0.934548
0.034961 0.568985 0.804938 0.648476 0.741854 0.484796 0.841530 0.274132 0.704010
0.405778
0.007990 0.776297 0.317005 0.660176 0.457447 0.060208 0.003451 0.057098 0.325871
0.513719
0.051990 0.174241 0.070493 0.367315 0.872369 0.219805 0.054544 0.673152 0.920530
0.791620
0.607699 0.955491 0.360605 0.412637 0.603966 0.102459 0.897433 0.445496 0.376591
0.601443
0.851274 0.384581 0.377740 0.168279 0.044757 0.835187 0.228487 0.048208 0.892285
0.554358
0.561927 0.944275 0.728599 0.632419 0.311589 0.600968 0.852224 0.366134 0.274119
0.772754
Child 2
0.902092 0.343041 0.124337 0.075700 0.909447 0.799654 0.417604 0.049726 0.699539
0.543107
0.692561 0.968319 0.976954 0.298890 0.437776 0.529632 0.268772 0.483003 0.817045
0.224225
0.087063 0.803600 0.494131 0.724064 0.134339 0.841439 0.903767 0.268440 0.470596
0.009896
```

```
0.117942 0.372688 0.352937 0.242279 0.448388 0.262383 0.041933 0.865992 0.312110
0.741472
0.409099 0.004671 0.709791 0.386053 0.303561 0.147567 0.915685 0.572332 0.630569
0.732730
0.796557 0.717632 0.536330 0.290688 0.441696 0.670669 0.132127 0.345462 0.939108
0.602723
0.355358 0.057050 0.975411 0.708295 0.299329 0.423799 0.970678 0.341262 0.289792
0.282788
0.082734 0.698890 0.287458 0.792525 0.084943 0.591019 0.940091 0.000628 0.163351
0.570660
0.733358 0.959908 0.288293 0.269688 0.250596 0.729989 0.940356 0.382723 0.075451
0.879465
0.985446 0.430809 0.936515 0.960857 0.139104 0.235843 0.384657 0.109782 0.577105
0.674448
Child 3
0.473834 0.189191 0.998444 0.698059 0.312348 0.931358 0.504224 0.136895 0.368024
0.418884
0.069356 0.380277 0.639466 0.025671 0.667475 0.780468 0.865718 0.762059 0.949442
0.535544
0.231281 0.295227 0.738972 0.991756 0.472445 0.795385 0.107613 0.646212 0.612873
0.130297
0.098869 0.086707 0.319487 0.097313 0.784767 0.631835 0.028671 0.288990 0.768730
0.396695
0.707875 0.838086 0.776972 0.347341 0.863757 0.444447 0.127809 0.729475 0.206505
0.077251
0.265019 0.437787 0.372478 0.003991 0.429543 0.844923 0.799376 0.537157 0.491135
0.412249
0.667453 0.590004 0.498956 0.986941 0.687316 0.283723 0.618776 0.715987 0.572713
0.387506
0.112682 0.280588 0.225592 0.889654 0.627929 0.089349 0.334101 0.755738 0.818823
0.540606
0.832989 0.083842 0.978393 0.205467 0.087833 0.407936 0.050391 0.887210 0.945093
0.541525
0.299459 0.612547 0.131529 0.798415 0.599487 0.818846 0.082138 0.218263 0.534833
0.654851
```

可见求得最大值的结果是正确的。

结论&其他

本实验中实现了openmpi程序的编写，进行了从环境搭建到结果调试等一系列过程，得到了正确的运行结果。

另外，openmpi默认不能够开过多的进程（slot），在4核笔记本上默认只能最多开4个slot，即使打开超线程后有8个逻辑处理器。但是仍然可以指定更多的进程，虽然这样可能导致性能下降。

创建一个hostfile，内容：

```
localhost slots=100
```

然后在运行时指定这个配置：

```
mpirun --hostfile hostfile -np 41 ./a.out
```

就可以使用更多进程。