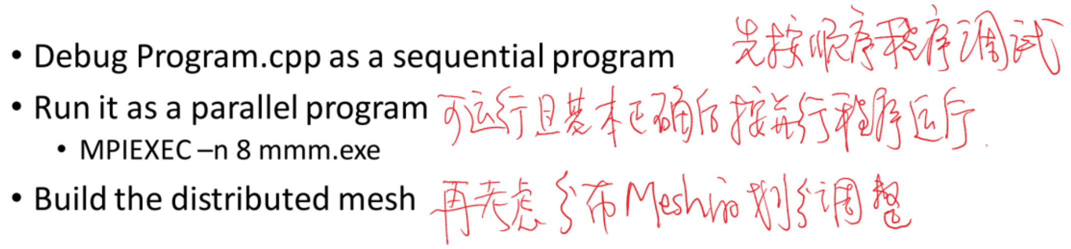
Mesh10 Parallel Program and Distributed Mesh

19335015 陈恩婷

1. 项目目的

参考课件和项目代码中的提示，修改程序中的错误。相关课件截图如下：



- Debug Program.cpp as a sequential program 先按顺序程序调试
- Run it as a parallel program 可运行且基本正确后按并行程序运行
  - MPIEXEC –n 8 mmm.exe
- Build the distributed mesh 再考虑分布Mesh的划分调整

2. 为实现目的存在的各种技术问题

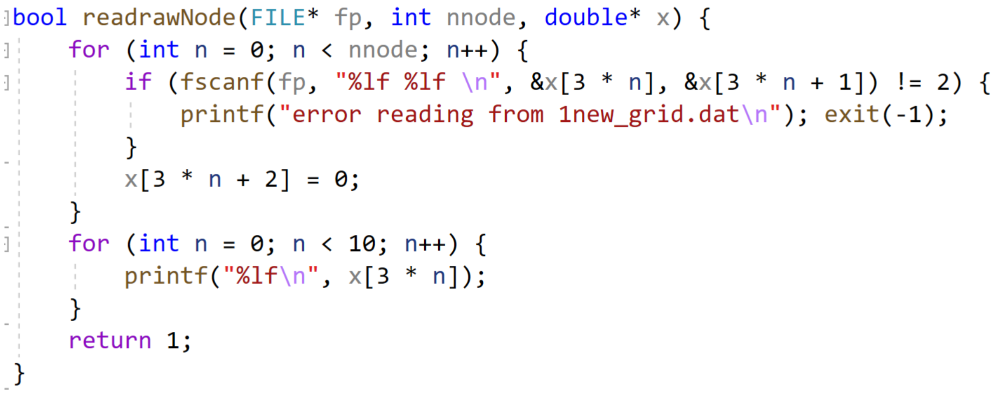
理解 mesh 的实现方法

熟练掌握 MPI 并行编程模型

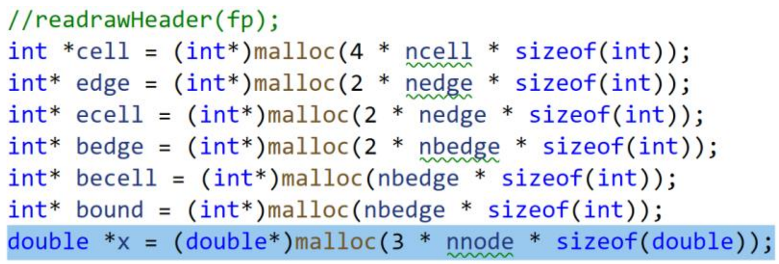
3. 用什么算法、数据结构、语言机制解决这些问题

通读整个程序的代码，找到可能存在问题的地方并尝试修复。

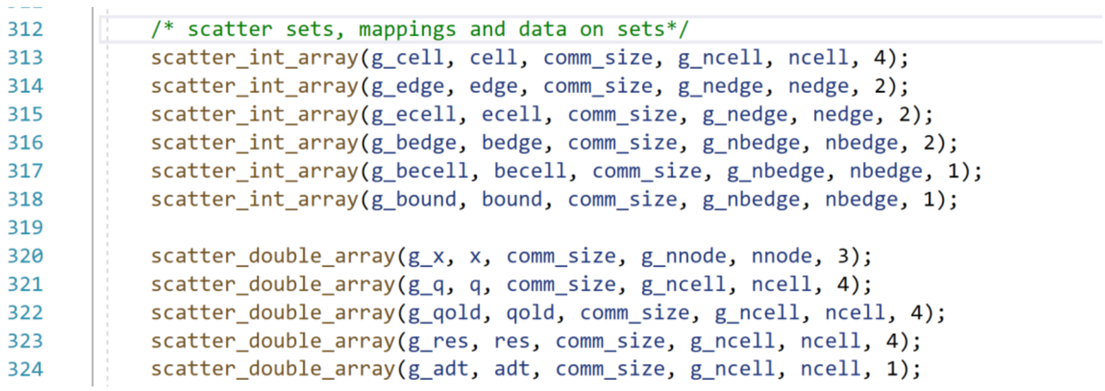
4. 对应的程序框架和实现代码



```
bool readrawNode(FILE* fp, int nnode, double* x) {
    for (int n = 0; n < nnode; n++) {
        if (fscanf(fp, "%lf %lf \n", &x[3 * n], &x[3 * n + 1]) != 2) {
            printf("error reading from 1new_grid.dat\n"); exit(-1);
        }
        x[3 * n + 2] = 0;
    }
    for (int n = 0; n < 10; n++) {
        printf("%lf\n", x[3 * n]);
    }
    return 1;
}
```



```
//readrawHeader(fp);
int *cell = (int*)malloc(4 * ncell * sizeof(int));
int* edge = (int*)malloc(2 * nedge * sizeof(int));
int* ecell = (int*)malloc(2 * nedge * sizeof(int));
int* bedge = (int*)malloc(2 * nbedge * sizeof(int));
int* becell = (int*)malloc(nbedge * sizeof(int));
int* bound = (int*)malloc(nbedge * sizeof(int));
double *x = (double*)malloc(3 * nnode * sizeof(double));
```

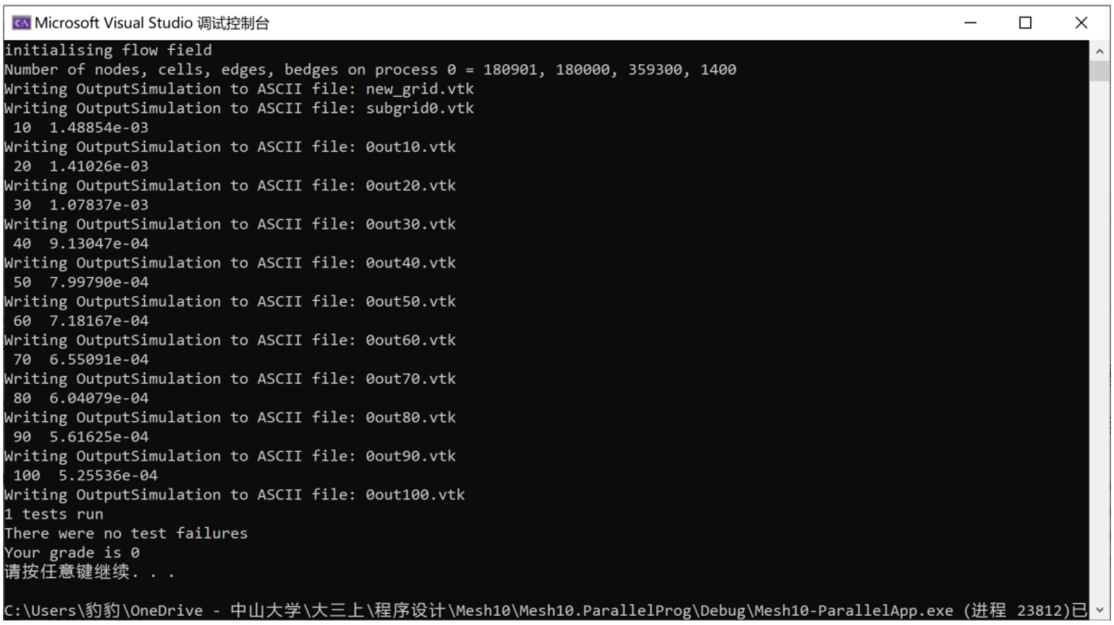


```
    /* scatter sets, mappings and data on sets*/
    scatter_int_array(g_cell, cell, comm_size, g_ncell, ncell, 4);
    scatter_int_array(g_edge, edge, comm_size, g_nedge, nedge, 2);
    scatter_int_array(g_ecell, ecell, comm_size, g_nedge, nedge, 2);
    scatter_int_array(g_bedge, bedge, comm_size, g_nbedge, nbedge, 2);
    scatter_int_array(g_becell, becell, comm_size, g_nbedge, nbedge, 1);
    scatter_int_array(g_bound, bound, comm_size, g_nbedge, nbedge, 1);

    scatter_double_array(g_x, x, comm_size, g_nnode, nnode, 3);
    scatter_double_array(g_q, q, comm_size, g_ncell, ncell, 4);
    scatter_double_array(g_qold, qold, comm_size, g_ncell, ncell, 4);
    scatter_double_array(g_res, res, comm_size, g_ncell, ncell, 4);
    scatter_double_array(g_adt, adt, comm_size, g_ncell, ncell, 1);
```

图中为已经修改好的代码。

5. 实验结果和结论



```
initialising flow field
Number of nodes, cells, edges, bedges on process 0 = 180901, 180000, 359300, 1400
Writing OutputSimulation to ASCII file: new_grid.vtk
Writing OutputSimulation to ASCII file: subgrid0.vtk
 10  1.48854e-03
Writing OutputSimulation to ASCII file: 0out10.vtk
 20  1.41026e-03
Writing OutputSimulation to ASCII file: 0out20.vtk
 30  1.07837e-03
Writing OutputSimulation to ASCII file: 0out30.vtk
 40  9.13047e-04
Writing OutputSimulation to ASCII file: 0out40.vtk
 50  7.99790e-04
Writing OutputSimulation to ASCII file: 0out50.vtk
 60  7.18167e-04
Writing OutputSimulation to ASCII file: 0out60.vtk
 70  6.55091e-04
Writing OutputSimulation to ASCII file: 0out70.vtk
 80  6.04079e-04
Writing OutputSimulation to ASCII file: 0out80.vtk
 90  5.61625e-04
Writing OutputSimulation to ASCII file: 0out90.vtk
 100  5.25536e-04
Writing OutputSimulation to ASCII file: 0out100.vtk
1 tests run
There were no test failures
Your grade is 0
请按任意键继续. . .

C:\Users\豹豹\OneDrive - 中山大学\大三上\程序设计\Mesh10\Mesh10.ParallelProg\Debug\Mesh10-ParallelApp.exe (进程 23812)已
```

如图所示，修正了串行程序的代码。

6. 对并行化原程序的一些想法

这次的程序若实现 MPI 的并行化，个人觉得会比较困难，原因是每个进程都需要修改 adt 和 res 等数组，而这些数组由于访问时都是需要借用其他数组先计算出坐标，再进

行访问，所以直接用多个进程运行原来的串行程序会在按下表划分这些数组后出现内存错误，但又很难做到比较方便地对它们进行划分。划分的困难性也就意味着并行化的方案并不是很显然，在做了一些考虑之后个人觉得若要实现比较直观的并行化可能需要重新建构整个程序的逻辑，就放弃了这一想法。