18329015 郝裕玮

Find\_bins函数解析：

/\*-------------------------------------------------------------------\*/

void Find\_bins(

      int bin\_counts[]      /\* out \*/,

      float local\_data[]    /\* in  \*/,

      int loc\_bin\_cts[]     /\* out \*/,

      int local\_data\_count  /\* in  \*/,

      float bin\_maxes[]     /\* in  \*/,

      int bin\_count         /\* in  \*/,

      float min\_meas        /\* in  \*/,

      MPI\_Comm comm){

   int i,bin;

   for (i = 0; i < local\_data\_count; i++){

      bin = Which\_bin(local\_data[i], bin\_maxes, bin\_count, min\_meas);

      loc\_bin\_cts[bin]++;

   }

   /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   MPI\_Reduce(loc\_bin\_cts, bin\_counts, bin\_count, MPI\_INT, MPI\_SUM, 0, comm);

   /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}  /\* Find\_bins \*/

MPI\_Reduce函数参数介绍：

int MPI\_Reduce(

void \*input\_data, /\*指向发送消息的内存块的指针 \*/

void \*output\_data, /\*指向接收（输出）消息的内存块的指针 \*/

int count，/\*数据量\*/

MPI\_Datatype datatype,/\*数据类型\*/

MPI\_Op operator,/\*规约操作\*/

int dest，/\*要接收（输出）消息的进程的进程号\*/

MPI\_Comm comm);/\*通信器，指定通信范围\*/

所以MPI\_Reduce函数的作用为：将所有的发送信息进行同一个操作。

在本题中的作用为：将每个进程中的loc\_bin\_cts中的每个同一位置的元素进行整体求和（MPI\_SUM），并将其输出消息发送至bin\_counts，且最后将bin\_counts中的消息发送至进程0.

Which\_bin函数解析：

/\*-------------------------------------------------------------------\*/

int Which\_bin(float data, float bin\_maxes[], int bin\_count,

      float min\_meas) {

   /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   int down=0;//bin标号的下界

   int up=bin\_count-1;//bin标号的上界

   int mid;

   float bin\_max,bin\_min;//用来记录某一个bin的上下界

   while(down<=up){//开始进行二分查找

      mid=(up+down)/2;

      bin\_max=bin\_maxes[mid];//当前查找的bin的上界

      if(mid==0){//特殊情况，若当前bin为0号bin

            bin\_min=min\_meas;

     }

     else{//通常情况下

         bin\_min=bin\_maxes[mid-1];//当前查找的bin的下界

     }

      if(data>=bin\_max){

            down=mid+1;//若数据大于当前bin[mid]的最大值，则down=mid+1

     }

      else if(data<bin\_min){

            up=mid-1;//若数据小于当前bin[mid]的最小值，则up=mid-1;

     }

      else{

            return mid;//若数据在bin[mid]的上下界范围之内，则找到，返回当前bin的标号，即mid

     }

   }

   /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   printf("Uh oh . . .\n");//二分查找找不到则打印该信息

   return 0;

}  /\* Which\_bin \*/

该函数中的补充部分主要是对数据data所属于的bin进行二分查找以归类，解析可见代码注释。

结果展示（在超算习堂的自由编程平台上编译运行）：

输入为：

Enter the number of bins：10

Enter the minimum measurement：0

Enter the maximum measurement：5

Enter the number of data：100

