课程名称：超级计算机组成原理

年级&班级	19 计科超算	专业(方向)	
学号	18324034	姓名	林天皓

一、 实验题目

使用 MPI 实现附件 PDF 文件中 2.7.1 节描述的直方图程序，进程 0 读取输入的数据，并将它们分配到其余进程，最后进程 0 打印该直方图。

请根据提供的代码，将代码补充完整！ 需要补充的部分见注释 PLEASE ADD THE RIGHT CODES 部分。(详细内容请下载附件 mpi_histogram.zip 查看)

二、实验内容

1 实验原理

本次实验是采用 MPI 编程的第一次实验，要求补全一个统计直方图的程序。MPI 是一种基于消息传递的并行编程技术,而不是一种具体的编程语言。 MPI 程序与 OpenMP 程序的最大不同就是 MPI 程序不仅可以适用多线程的方式并行运算还可以让程序以多进程的方式执行，以这种方式执行的程序并不共享内存，各个进程是通过消息传递来进行通信的。

通过阅读基础源代码，代码中实现统计直方图的程序由以下几个部分完成:

一、初始化

```
    MPI_Init(NULL, NULL);
comm = MPI_COMM_WORLD;
MPI_Comm_size(comm, &comm_sz);
MPI_Comm_rank(comm, &my_rank);
```

包括 MPI 的初始化，创建 MPI 的通信对象 comm，通过 MPI _Comm_rank 获取当前进程的 rank 函数。通过 MPI_Comm_size 确定与通信器相关联的组大小。接下来初始化变量，在 Get_input 函数中，先将用户输入通过 scanf 读取到 rank0 的进程上，然后通过 MPI_Bcast 将 rank0 获取的数据发送到所有进程上。

```
if (my_rank == 0) {
   printf("Enter the number of bins\n");
```

```
      scanf("%d", bin_count_p);
      printf("Enter the minimum measurement\n");
      scanf("%f", min_meas_p);
      printf("Enter the maximum measurement\n");
      scanf("%f", max_meas_p);
      printf("Enter the number of data\n");
      scanf("%d", data_count_p);
   }

   MPI_Bcast(bin_count_p, 1, MPI_INT, 0, comm);
   MPI_Bcast(min_meas_p, 1, MPI_INT, 0, comm);
   MPI_Bcast(max_meas_p, 1, MPI_INT, 0, comm);
   MPI_Bcast(data_count_p, 1, MPI_INT, 0, comm);
```

为每个进程申请空间

```
   bin_maxes = malloc(bin_count*sizeof(float));
   bin_counts = malloc(bin_count*sizeof(int));
   loc_bin_cts = malloc(bin_count*sizeof(int));
   data = malloc(data_count*sizeof(float));
   local_data = malloc(local_data_count*sizeof(float));
```

二、设置每个桶的最大值：

源程序中，通过 Set_bins 函数中，先对 rank=0 的进程计算一遍的最大值之后，通过 MPI_Bcast 发送到所有的进程上。

```
   if (my_rank == 0) {
      int i;
      float bin_width;
      bin_width = (max_meas - min_meas) / bin_count;

      for (i = 0; i < bin_count; i++) {
         loc_bin_cts[i] = 0;
         bin_maxes[i] = min_meas + (i+1)*bin_width;
      }
   }
   // set bin_maxes for each proc
   MPI_Bcast(  , bin_count, MPI_FLOAT, 0, comm);
   // reset loc_bin_cts of each proc
   MPI_Bcast(loc_bin_cts, bin_count, MPI_INT, 0, comm);
```

三、生成随机数据

在进程 0 上生成所有的随机数据，然后通过 MPI_Scatter 将随机数据均等的分发到各个子进程中。

```
   if (my_rank ==0) {
      data = malloc(data_count*sizeof(float));
      srandom(1);
      for (i = 0; i < data_count; i++)
         data[i] =
            min_meas + (max_meas - min_meas)*random()/((double) RAND_MAX);
#     ifdef DEBUG
      printf("Generated data:\n   ");
      for (i = 0; i < data_count; i++)
         printf("%.3f ", data[i]);
      printf("\n\n");
#     endif
      MPI_Scatter(data, local_data_count, MPI_FLOAT, local_data,
            local_data_count, MPI_FLOAT, 0, comm);
      free(data);
   } else {
      MPI_Scatter(data, local_data_count, MPI_FLOAT, local_data,
            local_data_count, MPI_FLOAT, 0, comm);
   }
```

MPI_Scatter 的实现示意图如下，主要用于将任务分解，下发给自进程。

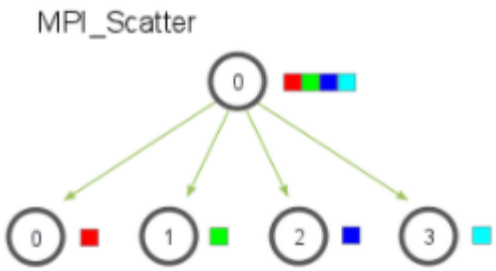

图 1-MPI_Scatter 示意图

三、Find_bins 函数即为每个子进程将 到的随机数据分到桶内的过程。其中作业的任务需要我们实现 Which_bin 函数查找 data 对应的桶内，并汇总结果。

四、最后需要通过 Print_histo 在 rank0 的进程上输出最终的结果。

2. **实验步骤**

一、首先实现 Which_bin 函数

```
int Which_bin(float data, float bin_maxes[], int bin_count,
      float min_meas) {
   /*************************************************************/
      for(int i=0;i<=bin_count;i++){
         if(data<bin_maxes[i]){
```

```
            return i;
          }
        }
    /*************************************************************/
    printf("Uh oh . . .\n");
    return 0;
}  /* Which_bin */
```

通过判断 data 最先小于哪一个桶的最大值，我们可以将对应桶的统计数量+1。

二、需要实现结果的汇总功能。

由于最后的 print 功能仅仅在 rank0 号进程上输出，这里我们需要把计算的到的桶数据合并到 rank0 号进程上。通过对 MPI 的学习，我们可以用多种方法完成这个步骤。

在 MPI 中有一类操作为 Collective communication，包括了 MPI_Bcast，MPI_Scatter，MPI_Gather，MPI_Allgather，MPI_Reduce，MPI_Allreduce。在这里可以使用 MPI_Reduce 或者 MPI_Allreduce 实现。

MPI_Reduce：

```
    /*************************************************************/
    MPI_Reduce(loc_bin_cts,bin_counts,bin_count,MPI_INT,MPI_SUM,0,comm);
    /*************************************************************/
```

MPI_Allreduce：

```
    /*************************************************************/
    MPI_Allreduce(loc_bin_cts,bin_counts,bin_count,MPI_INT,MPI_SUM,comm);
    /*************************************************************/
```

Allreduce 与 reduce 不同的是不需要指定汇总的进程 rank。

下面开始运行程序实验

四、实验结果

Reduce 版本：

选取 4 个桶，最小元素 200，最大元素 800，元素个数 200，输出结果如下。

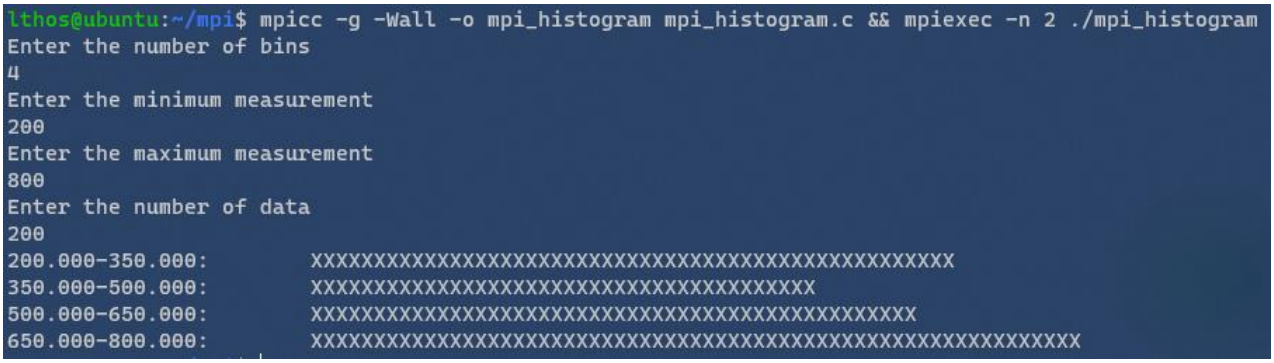

图 2-MPI_Reduce 版本结果

Allreduce 版本：

选取 4 个桶，最小元素 100，最大元素 600，元素个数 150，输出结果如下。

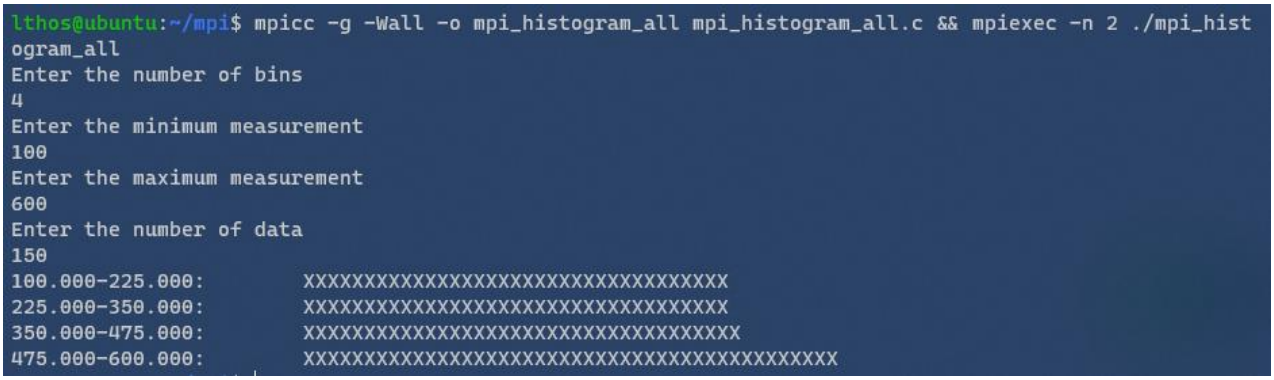

图 3-MPI_Allreduce 版本结果

经过上述实验，添加后的代码可以执行预定的随机数据的桶统计功能。

五、实验感想

本次实验是使用 MPI 编写程序的第一次实验，在本次实验中，安装并使用了 mpich3.3 进行并行程序的编译和运行。Mpi 通过进程间通信来同步数据，提供了多种通信函数，这一次实验中我们使用了 bocast 函数和 scatter 函数同步和分发数据，使用 reduce 和 allrecude 完成对计算完成结果的回收。

Mpi 中设计到的程序设计原理非常多，经过一段时间的了解与学习，其中的 Allreduce 函

数在最近的五年之间都发生了很多的改进。例如 allreduce 针对不同的数据规模采用了不同的通信方式，对于基础的小数据，使用了蝶形算法，每个节点同时接受和发送数据，充分使用了带宽。

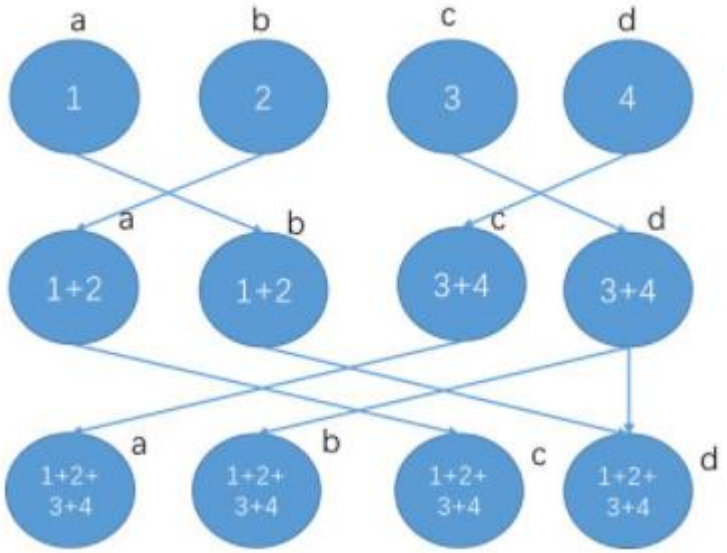

图 4-蝶形 reduce

而对于较大规模的数据，使用另一种 Ring allreduce 的方法，通过一环紧跟一环的数据沟通最终循环 n 轮完成数据的交换，采用这种方法可以减小数据的单次发送量。这和我们在计算机网络中学习的分组交换降低每次数据量和概念不谋而合。

图 5-环形 reduce

在更多更大结构的分布式网络中，在网格中进行 reduce 操作，对整个网格进行一次水平的 reduce 和一次垂直的 reduce 即可完成 reduce。

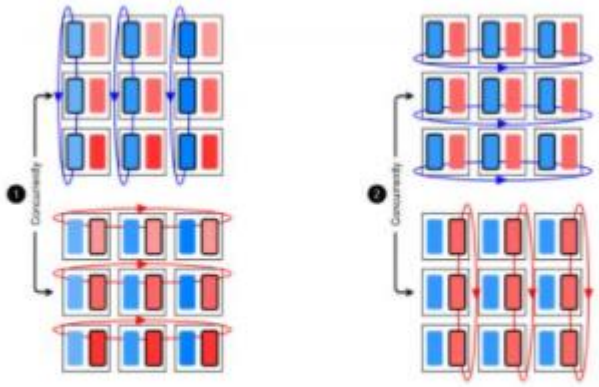

Figure 4: [best viewed in color] 2-D all-reduce across a hypothetical 3×3 torus. In the first phase (left), the first half of the tensors (blue) are summed along the vertical dimension while the second half of the tensors (red) are summed concurrently along the horizontal dimension. In the second phase (right), the dimensions are flipped, which completes the all-reduce for both halves.

图 6-2018 年提出网格 recude

通过以上的了解，在分布式计算网络发展日新月异的今天，每年仍然有大量的新技术运用到实践之中，只有紧跟时代前沿发展才能运用好超级计算机的强大能力。