Homework10

cat1997

November 2019

1 Task1

```
compile with the following:  mpicc -o run task2/src/mpiCentroids.c
mpiexec -n 4 ./run data/mpiCentroids/dat

/*As a Hokie, I will conduct myself with honor and integrity at all times.
 I will not lie, cheat, or steal, nor will I accept the actions of those who do.*/

#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>

float* readFirstColumn(const char* fname, int* Np);

int main(int argc, char **argv){
    // Initialize MPI library comms
    MPI_Init(&argc, &argv);

    // Get index of current process
    int rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    // Check for command line parameter
    if(argc < 2){
       printf("Please provide a filename.\n");
       exit(0);
    }
    if(rank < 0){
      printf("bad rank\n.");
      exit(0);
    }

    char* fname = argv[1];

    // Amount of data isn't known yet
```

```
int dataN;

float *myData;

// Message tags
int dataTag = 456;
int lenTag = 123;

if(rank==0){
    // Rank 0 reads the file and computes min
    myData = readFirstColumn(fname, &dataN);
    int dest = 1;
    MPI_Request outRequestLen;
    MPI_Request outRequestData;
  //send dataN
  MPI_Isend(
     &dataN, //data
     1, //len data
     MPI_INT,//type
     dest,
     lenTag,
     MPI_COMM_WORLD,
     &outRequestLen
    );

  //send data array
  MPI_Isend(
     myData, //data
     dataN, //len data
     MPI_FLOAT,//type
     dest,
     dataTag,
     MPI_COMM_WORLD,
     &outRequestData
    );

    // Find and print minimum value in data
    float lo = myData[1];
    for(int i=2;i<dataN;++i){
      if(myData[i] < lo){
         lo = myData[i];
      }
    }

    printf("Minimum: %f\n",lo);
}
```

```
    if(rank==1){
      // Rank 1 gets data from rank 0 and computes max
      int source = 0;
      MPI_Status lenStatus;
      MPI_Status dataStatus;

      //recieve dataN
      MPI_Recv(
          &dataN,
          1,
          MPI_INT,
          source,
          lenTag,
          MPI_COMM_WORLD,
          &lenStatus
          );

      float *inBuffer = (float*) calloc(dataN, sizeof(float));
      //receive data
      MPI_Recv(
          inBuffer,
          dataN,
          MPI_FLOAT,
          source,
          dataTag,
          MPI_COMM_WORLD,
          &dataStatus
          );
      //free data after it's all sent over
      free(myData);

      // Find and print maximum value in data
      float hi = inBuffer[1];
      for(int i=2;i<dataN;++i){
        if(inBuffer[i] > hi){
          hi = inBuffer[i];
        }
      }
      //free inBuffer
      free(inBuffer);
      printf("Maximum: %f\n",hi);
    }
    // Shut down MPI library comms
    MPI_Finalize();
    return 0;
}
```

```
float* readFirstColumn(const char* fname, int* Np){
    /*
    Reads the first column of data in formatted file
    File length is stored at Np
    */
    FILE *fp = fopen(fname, "r");
    float* data;
    int i;

    if(fp==NULL){
        printf("Failed to open file named: %s\n", fname);
        exit(0);
    }
    char buf[BUFSIZ];

    // Skip header
    fgets(buf, BUFSIZ, fp);

    // Read number of data points
    fgets(buf, BUFSIZ, fp);
    sscanf(buf, "%d", Np);

    // Skip header
    fgets(buf, BUFSIZ, fp);

    data = (float*) calloc(*Np, sizeof(float));

    // Initialize data set
    for(int n=1 ; n<*Np; ++n){
        fgets(buf, BUFSIZ, fp);
        sscanf(buf, "%f ",data+n);
    }
    return data;
}
```

2 Task 2

```
/*As a Hokie, I will conduct myself with honor and integrity at all times.
 I will not lie, cheat, or steal, nor will I accept the actions of those who do.*/
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
#include "data.c"
```

```
int find_max_clusters(data_t data);

  int main(int argc, char **argv){
  MPI_Init(&argc, &argv);

  int rank, size;
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);

  FILE *fp = fopen(argv[1], "r");
  int dataN;
  char buf[BUFSIZ];
  fgets(buf, BUFSIZ, fp);
  fgets(buf, BUFSIZ, fp);
  sscanf(buf, "%d", &dataN);

  float sumx;
  float sumy;

  //printf("size = %d\n", size);
  data_t data = dataRead(fp, rank, size, dataN);
  //printf("rank = %d, numpoints=%d\n", rank, data.N);

  int num_c = find_max_clusters(data)+1;
  int global_max;
  MPI_Allreduce(&num_c, &global_max, 1, MPI_INT, MPI_MAX,
              MPI_COMM_WORLD);

  //find local max of points
  data.sumx =  (float*) calloc(global_max, sizeof(float));
  data.sumy =  (float*) calloc(global_max, sizeof(float));
  for(int j=0; j<data.N;j++){
      data.sumx[data.cluster[j]] = data.sumx[data.cluster[j]]+data.x[j];
      data.sumy[data.cluster[j]] = data.sumy[data.cluster[j]]+data.y[j];
  }

  float* global_sumx =  (float*)calloc(global_max, sizeof(float));
  float* global_sumy =  (float*)calloc(global_max, sizeof(float));

  //find global sum from reduction of all sums
  MPI_Allreduce(data.sumx, global_sumx, global_max, MPI_FLOAT,
                MPI_SUM, MPI_COMM_WORLD);
  MPI_Allreduce(data.sumy, global_sumy, global_max, MPI_FLOAT,
                MPI_SUM, MPI_COMM_WORLD);

  if(rank ==0){
    for(int i=0;i<global_max;i++){
      float avgx = global_sumx[i]/global_max;
      float avgy = global_sumy[i]/global_max;
      printf("%f %f\n", avgx, avgy);
    }
  }
  free(global_sumx);
  free(global_sumy);
  free(data.sumx);
  free(data.sumy);
  free(data.x);
  free(data.y);
  free(data.cluster);

  MPI_Finalize();
  return 0;
}

int find_max_clusters(data_t data){
  int max = 0;
  for(int i=0; i<data.N;i++){
    if(data.cluster[i] > max){
      max = data.cluster[i];
    }
  }
  return max;
}

data.c:
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

// struct to hold data set
typedef struct{
  // number of data points
  int N;

  // coordinates of data points
  float *x;
  float *y;
  int *cluster;

  float* sumx;
  float* sumy;
```

```
}data_t;


// read data from file
data_t dataRead(FILE *fp, int rank, int size, int dataN){
  data_t data;
  char buf[BUFSIZ];

  int num_points;
  if(dataN%size != 0 && rank == size-1){
    data.N =  dataN/(size-1) + dataN%(size-1);
  }
  else{
    data.N = dataN/(size-1);
  }
  data.x = (float*) calloc(data.N, sizeof(float));
  data.y = (float*) calloc(data.N, sizeof(float));
  data.cluster = (int*) calloc(data.N, sizeof(int));
  // initialize data set
  for(int n=0;n<data.N;++n){
    fgets(buf, BUFSIZ, fp);
    sscanf(buf, "%f %f %d", data.x+n, data.y+n, data.cluster+n);
  }

  return data;
}
```

```
cat@cat-VirtualBox:~/cmda3634/HW10/task2$ mpiexec -n 4 ./run data/mpiClusters.dat
0.158856 0.173772
13.098998 0.140976
26.657551 0.088010
39.531525 0.050308
53.561890 0.070828
65.517281 0.126007
79.171638 0.031836
95.477890 0.160451
109.928360 0.040554
119.999153 0.100595
0.100015 13.136867
13.457556 13.443953
26.493523 13.348011
39.568241 13.306907
53.915688 13.658408
67.617928 13.622119
79.704689 13.472793
93.694618 13.417995
```

```
108.906837 13.772530
119.564339 13.377163
0.102903 26.713675
13.606022 26.916086
27.329050 27.320206
40.215019 26.889700
53.193790 26.685862
66.477783 26.685984
81.980461 27.378864
92.426689 26.497297
106.824699 26.774755
118.794579 26.584993
0.142679 39.712135
13.293527 39.665321
27.247002 40.933720
39.839470 39.812592
53.873344 40.394928
67.736328 40.718678
79.683495 39.916107
92.539970 39.673088
104.348106 39.096535
123.388542 41.275341
0.122439 52.939175
13.408434 53.099743
27.571899 54.962387
39.829926 53.202518
53.397938 53.428585
66.099274 52.784634
78.790352 52.628021
93.654900 53.554115
108.585350 54.387806
121.956543 54.314342
0.100740 67.171883
13.150398 65.363197
26.595520 66.178566
40.338036 67.171265
53.803078 67.245308
68.285339 68.150993
81.118889 67.703285
90.860909 64.986496
107.418831 67.143463
114.412750 63.594410
0.086671 80.434578
13.291352 79.014816
27.577053 82.541191
39.069988 77.986794
```

```
53.145889 79.543793
65.801590 78.887291
79.766853 79.838837
95.544769 81.898674
106.985672 80.304016
123.473488 82.329071
0.148372 96.377419
13.564683 94.023430
26.102497 90.902901
40.865463 95.147011
54.456841 95.070389
67.764252 94.792145
79.605408 92.817177
93.519363 93.501366
108.276344 94.784698
117.329811 91.347771
0.163240 108.297569
13.710499 108.730019
26.573227 106.058441
```