Cmpe 300 Programming Project
Fall 2010
Parallel k-Means Algorithm

Due Date: 10/01/2011 17:00

1 Introduction

In this project you are going to implement a parallel *C* algorithm using *MPI (Message Passing Interface)* classes. The algorithm to be implemented is a simple machine learning algorithm to divide given data points into groups according to a distance measure. In our project we will use 2D cartesian points and Euclidean distances.

2 The k-Means Algorithm

The k-Means algorithm aims to partition N data points into k groups called *clusters*, such that each data point belongs to a cluster with nearest mean. A visual description of the problem is given in Figures 1 and 2.

Figure 1: Input: 100 points on Cartesian Grid

3 Basic Idea of the k-Means Algorithm

It's an iterative algorithm which runs until it converges to the solution. The algorithm first randomly selects the *centroids*. In each iteration the algorithm first determines the cluster membership of each data point according to the its distances to the centroids. Then the centroid

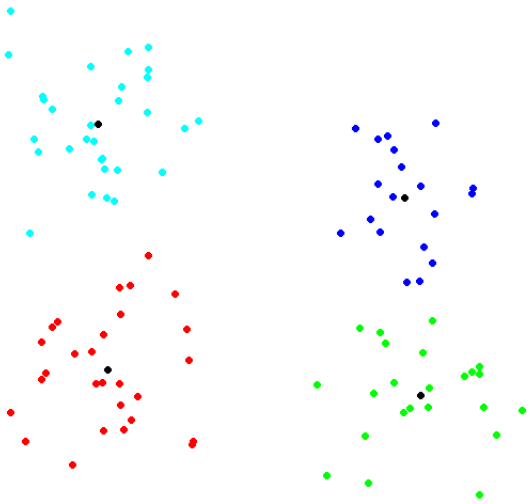

Figure 2: Output: Result of the k-Means Algorithm

of each cluster is updated as the mean of the points that belongs to that cluster. The iterations stop either at a given time step, or whenever the convergence occurs. The algorithm is said to be converged whenever the cluster memberships do not change anymore. Pseudocode is given in Algorithm 1.

3.1 Pseudocode for k-Means

Algorithm 1 Sequential k-Means Algorithm

Require: *data points*: 2D cartesian points.
Require: *number of clusters*
Ensure: *centroids*: cluster centers
Ensure: *cluster memberships*: the numbers showing the cluster assignments of data points
  Initialize *centroids* randomly.
  Initialize *cluster memberships* to dummy values.
  while not converged **do**
    Find the *cluster memberships* of data points by comparing their distances to the *centroids*.
    if *cluster memberships* did not change **then**
      break
    else
      Update all *centroids*
    end if
  end while

4 Parallel k-Means

Assume that you have $N + 1$ processors, such that one of them is the master. The master processor reads the data points, which are the (X, Y) pairs, and the number of clusters k from a text file. No other processor can make read/write operations on files. The role of the master

is to divide the data into N, and send each portion to a processor. The processors will calculate the cluster memberships, and master will calculate the centroids.

1. Master reads data and divides it into N portions and sends one of them to each processor.
2. Master randomly initializes centroids.
3. Master sends all of the centroids to the all of the processors.
4. Each processor calculates the cluster membership of the data points assigned to it, and sends the results back to the master.
5. Master calculates new cluster centers by taking the means.
6. If converged program terminates, otherwise the algorithm continues from step 3.

5 An example run

Assume that you are given 100 points, and 6 processors (1 of them is the master), and the number of the clusters is 2 . The master will read the data, and divide it into 5, and send 20 points to each of the 5 processors. Then it will select 2 centroids randomly in the appropriate range (in our case this range is $[0, 1]$). This centroid values will be send to each of the 5 processors. These processors will find the cluster memberships of the 20 data points sent by the master. The results will be sent back to master as a 1×20 array. The master will first check the convergence. If converged the master will write the output and the program will stop. Otherwise the master will update the centroids and send them to other 5 processors and iterations will continue.

6 Input and outputs

Your program will read the data points and the number of clusters from a text file. The range of the data points are $[0, 1]$. The results will be written in a text file by the master processor (remember that only master can read/write files). The output will be cluster centers, and the cluster memberships of the points. The format of the files are:

6.1 Input File Format

```
NUM_CLUSTERS
NUM_DATA_POINTS
POINT_1_X,POINT_1_Y
POINT_2_X,POINT_2_Y
...
...
POINT_2_N,POINT_N_Y
```

6.2 Output File Format

```
NUM_CLUSTERS
NUM_DATA_POINTS
CENTROID_1_X,CENTROID_1_Y
CENTROID_2_X,CENTROID_2_Y
```

```
...
...
CENTROID_k_X,CENTROID_k_Y
POINT_1_X,POINT_1_Y,CLUSTER_VALUE
POINT_2_X,POINT_2_Y,CLUSTER_VALUE
...
...
POINT_2_N,POINT_N_Y,CLUSTER_VALUE
```

6.3 How to run the program

You should read the names of the input and output files as the arguments to your main. Furthermore, your program should be able to run with any number of processors. Your program will be tested with the following script:

```
mpiexec -n NUM_PROCESSORS ./your_project INPUT_FILE OUTPUT_FILE
```

7 Test Data and Validation

7.1 Test cases from the course web page

You can find 3 test cases on course web page, with both input and output files. There is also a visualization tool on the course web page. You can upload your output files and if they are saved in the given format, the tool will show your cluster centers, and the data points in different colors according to their cluster values.

7.2 Your own test case

You are supposed to make your own test cases. You can create it simply by randomly selecting 100 2D points in the range $[0, 1]$. You are supposed to create 3 test cases, and try them with different number of clusters and different number of processors.

8 Final Remarks

- Write your code with meaningful comments. Especially messaging parts should be commented.
- Include results of your own test cases in your report.
- Submit your C code as a single file with the format STUDENT_NUMBER.c
- The project will be done individually, not as a group.
- The deadline will not be extended.
- For the MPI environment and tutorials check the course web page.
- See http://www.cmpe.boun.edu.tr/ gungort/informationstudents.htm for general information (submission process, documentation, etc.) about programming projects.