Student Name: Mehmet Yasin ŞEREMET
Student ID: 2017400102

CMPE300 MPI PROJECT

Introduction

The main aim of this project is to experience the MPI library. While using MPI, the concept is to implement the relief algorithm in many processes parallelly on equally distributed data. Thus, we obtain the important features and eliminate the unnecessary features from a given dataset via benefiting from the opportunities of parallel programming.

Program Execution

Compile and run commands are the same as specified in the description.

As follows,

```
> mpic++ -o cmpe300_mpi_2017400102 ./cmpe300_mpi_2017400102.cpp

> mpirun --oversubscribe -np <P> ./cmpe300_mpi_2017400102 <inputfile>
```

My Open MPI version is 4.0.3.

My Ubuntu version is 20.04.1 LTS.

Program Structure

Firstly, my program uses mpi, iostream, stdlib, sstream, fstream, unistd, bits/stdc++ libraries.

My program consists of only main function. So, I explain the procedure from the top on down.

After I write the required MPI start functions, firstly, my program declares the variables for the number of total processes, instances, features, iterations, and top features and initializing them to values which are given in the input text file.

Nextly, the array including all input data which is distributed by master to each slave named as arrSent[ ][ ][ ] and the second array including data which is received by each slave named as arrReceived[ ][ ] are declared. Each slave only receives its part of the data in this received array.

arrSent[ ][ ][ ] is a 3-dimensional array. First dimension indicates rank of slave which the related data will be sent to. Second dimension indicates the instance number or line will be sent to related slave. Third dimension indicates the feature number or class label.

arrReceived[ ][ ] is the array received by each slave. It is 2-dimensional. First dimension indicates the instance number will be received by related slave. Second dimension indicates the feature number or class label.

Then, 3 arrays about weights of features names as weight[ ], top features of a slave named as slaveTopFeatures[ ], top features of the master named as masterTopFeatures[ ] are declared here.

Student Name: Mehmet Yasin ŞEREMET
Student ID: 2017400102

My program starts the first part of the master process here. In this part, all the data in the input file is read and put in the arrSent[ ][ ][ ] array as mentioned above. From now, each part of arrSent[ ][ ][ ] consists of equally distributed amount of data to be processed by slaves.

By using MPI_Scatter function in the library, arrSent[ ][ ][ ] scatters part of the data belonging each slave to its arrReceived[ ][ ] array. From now, each slave has its data to process in its arrReceived[ ][ ] array.

There is a masterSignal and a while loop to finish the program successfully. From now, all the procedures are done in this while(masterSignal) loop.

The section for slave process starts here. This is the longest part of the program.

Firstly, there is slaveTopFeatures[ ] array initialization of its elements to -1 to ease the job.

Then, the main algorithm of the project, Relief Algorithm, starts here. There is an array named as target[ ] to keep the target instance and it is initialized according to description. There is 2 other arrays named as distArrHit[ ] and distArrMiss[ ]. As it could be understood from their names, they keep the values of distances between the target instance and hit&miss instances. (Distances calculated via Manhattan distance formula as it is specified.) The calculation for distances is done next. Now, we have all distances kept in these arrays.

In a basic loop, the program finds the indexes(or line numbers) of nearest hit&miss instances with smallest distance to the target instance. The result indexes are kept in resultHit and resultMiss variables.

After that, diff(A,Ri,H)/m and diff(A,Ri,M)/m are calculated by using the resultHit and resultMiss. Their values are put in the weight formula and results are stored in weight[ ] array. So, we have weight[ ] array of features now. The relief algorithm ends here.

Now, indexes(or line numbers) of given number of top features having largest weight in the weight[ ] array are extracted and stored in slaveTopFeatures[ ] array. After sorting this array for true output format, each slave prints its top features. The part of slave processes ends here.

By using MPI_Gather function in the library, each slave sends its slaveTopFeatures[ ] array to the master's masterTopFeatures[ ] array.

Now, the last part of master process starts here. After removing duplicates and sorting the remaining list of top features gathered from slaves. Only, the results for master are printed as specified output format.

At the end, masterSignal is broadcasted to end all processes by using MPI_Bcast.
MPI_Finalize ends the MPI section, the program terminates successfully.

Difficulties Encountered

The main difficulty I encountered was the problems while trying to create helper functions to ease reading the code. Since I used a 2-dimensional array arrReceived[ ][ ] as main data source in all of my calculations, I needed to pass it as a parameter to helper functions. But while passing a 2-dimensional function as parameter, first dimension of it must be declared as a constant integer in C++. Since I read the size of this dimension from the input file as given, it could not be a constant. Therefore, I write the all code in the main function. As an advantage, it was easier to implement.

Student Name: Mehmet Yasin ŞEREMET
Student ID: 2017400102

Conclusion

I think this was a joyful and instructive project to meet MPI and parallel programming. The main focus was on the parallel programming not on the complex algorithm. From this aspect, it was really successful. The concept was also really good. Relief is trend and up to date. Even though it is a small part of ML, implementing something that is close to reality was satisfying. Thank you.