Project- 2 Part-1

Prepared by Srinivasan Rajappa
Student ID: 50134691

1 SYNOPSIS

Here a MapReduce application which runs on Hadoop Framework counts the number of students using a Hall at University at Buffalo for every semester since the year 1931.

The input data (a .csv file) consists of information viz. Serial #, Room #, Semester, Capacity of a room, timings of the class etc. Moreover, the input data like any data source has incorrect data so this application also discards such entries.

2 MOTIVATION

In order to understand the dynamics of schedule management in any institution it is helpful if some statistical insights are available. In this report one aims to take a tiny step at analysis of data; one finds the enrollment of students in academic halls at UB. The final aim is to find out the overall utility served by the academic halls in the University. This will help in reasoning planning allotment and scheduling in future.

3 INPUT DATA

The input data was provided after scraping online data available in one of the websites of UB. The data was then condensed in form of .csv files. Each row in that file provided details on the number of students who enrolled, the capacity of a room, timings of the class, the name of the class, Semester along with year when it was conducted.

4 APPROACH

There were three phases viz. cursory analysis of input data, removal of inconsistent data entries and calculation. On examining the data some details emerged.

1. Each row consisted and uniquely defined a room for a particular semester and to how many students used for which days of a week.
2. Some rows had inconsistent data viz. Hall Name equivalent to ***Arr***, classes beginning ***before 8:00 AM***, Course named ***Unknown*** etc.
3. Some rows had more than 9 columns, the reason for such an occurrence was because of the introduction of "**,**" in between the names of course. Such introduction implied that there existed extra column according to the MS-Excel application that opened it.
4. There were several records and as a .csv file that consists only of text, the size of the file was approximately 45 MB.

The next phase was to successfully remove these inconsistent data. For the same in MapReduce program I decided to exclude any rows that had the aforementioned inconsistencies. I also decided to exclude the

data where the number of columns exceeded 9 columns. In the MapReduce application thus created, I decided to extract only those rows where the number of tokens created by method `StringTokenizer` equals quantity 9. Lines 19/20 in application code named `HallEnrollmentCounter.java` exhibits the same.

```
…
StringTokenizer itr = new StringTokenizer(value.toString(),",");
      if(itr.countTokens()==9){
…
```

Lines 40-43 try to avoid the rows which have entry of **Arr** or **Unknown** in the name of the building.

```
if(hall.toUpperCase().equals("UNKNOWN"))
   return;
if(hall.toUpperCase().equals("ARR"))
   return;
```

The next phase was to calculate these values in order display the output. The Mapper section of this program will be called for every row. For every row a preprocessing is done wherein inconsistent data is rejected. The correct values are identified are sent as a <Key, Value> pair. Here, a key is of the form ***<HALLNAME_SEMESTER, #ENROLLED>*** a HallName is extracted after removing the characters trailing after a single space. This was done easily by splitting the 3rd column whose token ID was 2. It is performed at line 39 as follows:

```
hall = hall.split(" ")[0];
```

Thus for example if the column had a value similar to ***Knox 225*** the result of the computation above will result to ***Knox*** **.**

The next operation was to concatenate the Semester entry (which is the 2nd column whose token ID is 1) with the hall name. So, the final operation where Semester is ***Fall 2014*** will yield result ***Knox_Fall 2014.*** Line 44 performs this operation.

```
String hall_Year = hall+"_"+year;
```

The number of enrolled students was easy to fetch as it was the 8th column with token ID 7. Subsequently I was able to save it as a key value pair with the following code at Line 46-47.

```
word.set(hall_Year);
context.write(word, new IntWritable(Integer.parseInt(enrolled)));
```

These map entries will later be used by Reducer code Lines 53-67. This is the same code that was used in the tutorial.

5 RESULT

The code was used to perform sample runs on sample input. I obtained around 1000 lines from the end of the data input and ran the application. The results were consistent. The file names are as follows:
1. Input: /sample/testFile1K

2. Output: /sample/result1K

In addition to performing the above diagnostics, I also performed the application run on the complete dataset on Amazon Web Services (AWS) using Elastic MapReduce. I followed the instructions provided in the tutorial.

I took three trials to recover the final output. The first time I supplied the .csv file which was provided to me in the beginning. It seems like the EMR or MR doesn't work when the input file is of the format .csv. In order to get the correct format, I changed file to normal file using the `cat` and `pipe` operation in linux terminal. As the size of the file was huge thus I provided the dropbox link to the file and didn't keep it with other files meant for submission.

```
>  cat bina_classschedule.csv > fileOK
```

The other two times, I just run it with correct parameters. After a waiting time of 33 minutes and 10 minutes for other two runs respectively. I got 7 result files, I added them to the list of submissions. The files could be found in the directory /output/part*

6 REFERENCES

1. The tutorials and the online documentation.
2. The lively student community and the helpful staff at Piazza.