COMP 6730 Advanced Database Systems Project

Nikhil Nagendra Swami
nikhil_swami@student.uml.edu
01539304
December 3, 2016

1) Overview:

For this project, I have configured the Hadoop environment as well as the Eclipse with Hadoop-eclipse-plugin, then setting up the namenode and other server on local machines, start the eclipse and create a Map-Reduce project in Eclipse.
Hadoop MapReduce is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.
The Map Task: This is the first task, which takes input data and converts it into a set of data, where individual elements are broken down into tuples (key/value pairs).
The Reduce Task: This task takes the output from a map task as input and combines those data tuples into a smaller set of tuples. The reduce task is always performed after the map task.
The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node.

2) Hadoop Project:

Main aim of the project is to use MapReduce programming to parse text files and extract required information performing different operations like Selection, Projection, Natural Join and Aggregation.

To implement all operations a procedure followed is:
1. Initialize Driver program to instantiate Mapper and Reducer jobs.
2. Input specific file to mapper job.
3. Parse the specified text file line by line, split the fields based on delimiter ',' and store in an array as different columns.
4. Mapper maps the required columns and send pair to reducer job.
5. Reducer receives pairs from all mapper jobs and perform calculations to output required results.
6. Results are written in to output file.

3) Selection Query:

Computing Selection by MapReduce:
Task: To find cities whose population is larger than 300,000.
Input File: city.txt
Output File: output.txt

In the map function each line of the “city.txt” are split on delimiter ‘,’ and individually stored in different columns. For this selection query we need column 2nd and 5th which are send as the key value pair(K, V) to the reducer.

In the reducer function all the key value pairs received from mapper are filtered. And based on the filter condition of population > 300,000 are written to the output.
Resulting rows are taken as output.

OUTPUT:

ans1.txt | ans2.txt | ans3.txt | ans4.txt

```
Abeokuta
Abidjan
Abu Dhabi
Abuja
Acapulco de Ju□rez
Accra
Adana
Addis Abeba
Adelaide
Aden
Ado-Ekiti
Agra
Aguascalientes
Ahmedabad
Ahome
Ahvaz
Ajmer
Akita
Akola
Alberton
Albuquerque
```

4) Projection

Computing Projection by MapReduce:
Task: To find all the name of the cities and corresponding district.
Input File: city.txt
Output File: output.txt

In the mapping function, the mapper reads each line of the city.txt which has text separated by delimiter ',' and stored in separate columns.
For this selection query we need column 2nd and 4th which are send as the key value pair(K, V) to the reducer.

In the reducer, it receives data from the mapper as key value pair and the columns are further filtered down based on the condition of finding all names of cities and corresponding district.
Resulting rows are taken as output.

OUTPUT:

ans1.txt | ans2.txt | ans3.txt | ans4.txt

```
A Coru□a (La Coru□a)    Galicia
Aachen  Nordrhein-Westfalen
Aalborg Nordjylland
Aba Imo & Abia
Abadan  Khuzestan
Abaetetuba  Par□
Abakan  Hakassia
Abbotsford  British Colombia
Abeokuta    Ogun
Aberdeen    Scotland
Abha    Asir
Abidjan Abidjan
Abiko   Chiba
Abilene Texas
Abohar  Punjab
Abottabad   Nothwest Border Prov
Abu Dhabi   Abu Dhabi
Abuja   Federal Capital Dist
Acapulco de Ju□rez  Guerrero
Acarigua    Portuguesa
Accra   Greater Accra
```

5) Natural Join:

Computing Natural Join by MapReduce: Task: To find all countries whose official language is English.
Input Files: country.txt and countrylanguage.txt
Output File: output.txt

In the map function each line of the “countrylanguage.txt” are split on delimiter ‘,’ and individually stored in different columns. For this selection query we need column 1st and 2nd columns which has the country code and respective language are send as the key value pair (K, V) to the reducer.

In the reducer, the key value pair of the country code and respective language send by the mapper are captured and stored. They are further used to find the name of the country where language is english using natural join between the 2 columns.
Resulting rows are taken as output.

OUTPUT:

ans1.txt | ans2.txt | ans3.txt | ans4.txt

```
Aruba
Anguilla
Netherlands Antilles
American Samoa
Antigua and Barbuda
Australia
Bahrain
Belize
Bermuda
Barbados
Brunei
Canada
Cocos (Keeling) Islands
Cook Islands
Christmas Island
Cayman Islands
Denmark
Falkland Islands
United Kingdom
Gibraltar
Guam
```

6) Aggregation:

Computing Aggregation by MapReduce:
Task: To find how many cities each district has.
Input File: city.txt
Output File: output.txt

Map:
In the map function each line of the “city.txt” are split on delimiter ‘,’ and individually stored in different columns. We get the city-district key value pair (K, V).

Reduce:
All the <key, value> pairs are received from all mapper jobs are used to count of number of cities is calculated based on received pairs.

OUTPUT:

ans1.txt ans2.txt ans3.txt ans4.txt

```
ARMM    1
Abhasia [Aphazeti]  1
Abidjan 1
Abruzzit    1
Abu Dhabi   2
Aceh    2
Acre    1
Adana   3
Addis Abeba 1
Aden    1
Adiyaman    1
Adygea  1
Adzaria [At□ara]    1
Afyon   1
Aguascalientes  1
Ahal    1
Aichi   15
Aimei   1
Ajman   1
Akershus    1
Akita   1
```

7) Problems Faced:

On of the problems faced in this project was the installation of Hadoop on linux system and using it to get the jar out file for processing of the map-reduce function.
Selection of the columns and getting the values by the delimiters gave some problem but once I was able to get for the first program, it was easier to get the values for processing.

8) Conclusion:

The main conclusion from this project was to get a sense of hadoop working and to process huge amount of data and selecting the required data to find according to the query processed. The assignments were pretty clear and understandable. Processing on the key value pair helped in processing the queries and getting the desired output.

9) Reference:

1. "Welcome To Apache™ Hadoop®!". Hadoop.apache.org. N.p., 2016. Web. 3 Dec. 2016.

2. "Hadoop - Command Reference". www.tutorialspoint.com. N.p., 2016. Web. 3 Dec. 2016.

3. "Setting Up A Apache Hadoop 2.7 Single Node On Ubuntu 14.04". the power of data. N.p., 2016. Web. 3 Dec. 2016.

4. "Word Count Job Implementation In Hadoop". YouTube. N.p., 2016. Web. 3 Dec. 2016.