CSC 643 – Big Data & Web Intelligence

$sudo Team1 -Ahmad/Sean

Project 2

Submission Date: October 28, 2019

Table of Contents

Task Ranking

- Member 1: **Ahmad Alshawaf**

  Handled 50% of the project from start to complete

  - Task 1: analyzing data
  - Task 2: solving problem 1
  - Task 3: solving problem 2
  - Task 4: solving problem 3
  - Task 5: organizing report

- Member 2: **Sean Sothey**

  Handled 50% of the project from start to complete

  - Task 1: analyzing data
  - Task 2: solving problem 1
  - Task 3: solving problem 2
  - Task 4: solving problem 3
  - Task 5: organizing report

Introduction

Data science has become an extremely rewarding career choice for people interested in extracting, manipulating, and generating insights out of large volumes of data. To fully leverage the power of data science, scientists often need to obtain skills in databases, statistical programming tools, and data visualizations. Companies surely need data scientists to help them empower their analytics processes, build a numbers-based strategy that will boost their bottom line, and ensure that enormous amounts of data are translated into actionable insights. In this project, we would like to introduce one of the most popular platform tools to analyze the data.

Apache Pig programs are written in a query language known as Pig Latin that is similar to the SQL query language. To execute the query, there is a need for an execution engine. The Pig engine converts the queries into MapReduce jobs and thus MapReduce acts as the execution engine and is needed to run the programs.

Apache Pig can be started either in cluster mode or in local mode. To start Apache Pig in local mode, the developers should use the option "-x local". If no option is specified, then Pig by default is started in the cluster mode. The cluster mode allows Pig to access data file present on HDFS, whereas in local mode only files within the local file system can be accessed.

Therefore, we are going to run the Apache Pig on HDFS as a model in this study course to fulfill the assigned "Project 2".

Analyzing Data

1) Data Description

Referring to the reference at the end of the report, in **2. Element Names and Definitions**, we can see that there are 12 data fields in this data set.

Here is the summary table of these data fields:

FIELD	START POS	END POS	ELEMENT NAME	VALUE
001	1	3	Record-Type	HPD
002	4	11	Station-ID	13136306
	4	5	State-Code	13
	6	9	Cooperative Network	1363
	10	11	Cooperative Network Division Number	06
003	12	15	Element-Type	HPCP
004	16	17	Element-Units	HT
005	18	21	Year	2013
006	22	23	Month	09
007	24	27	Day	0015
008	28	30	Number-Reported-values	004
009	31	34	Time-of-Value	0100
010	35	40	Data-Value	00160
011	41	41	FLAG1	
012	42	42	FLAG2	

Table 1: Data Identification

Now, let's observe this first three data values formats.

```
HPD01419307HPCPHT20130900240031700 00150  1800 00010  2500 00160

HPD01419307HPCPHT20130900300020400 00010  2500 00010

HPD01420904HPCPHT20130900010030100 00000g 1500 00010  2500 00010
```

We can see that the format of the data is combined together in each row, and the data value is counted after 2500, also including FLAG1 and FLAG2. In this case, we are going to chunk this data format into different column, using substring method.

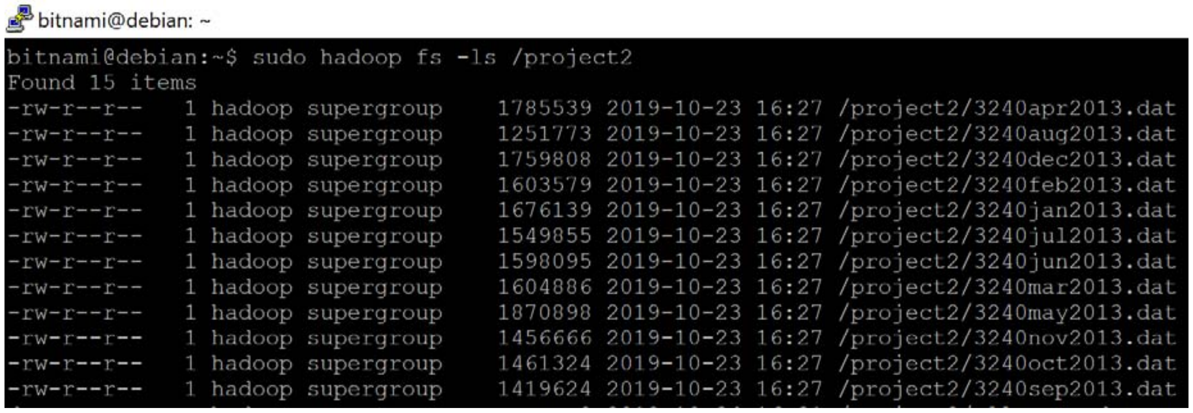
2) Rearrange Data Tuples

- **Step 1**: There are two locations to load the data in this course, one is from local machine, and the other one is from HDFS. For this project, as mentioned from the beginning, we chose to work on HDFS. Therefore, first we copy all the data files from local machine (Bitnami) to HDFS, into a directory named "project2", using the following command:

  `$sudo hadoop fs -put '/home/bitnami/project2/' '/project2'`

  Check if the copied files exist in HDFS directory, using the following command:

  `$sudo hadoop fs -ls /project2`



```
bitnami@debian: ~
bitnami@debian:~$ sudo hadoop fs -ls /project2
Found 15 items
-rw-r--r--   1 hadoop supergroup    1785539 2019-10-23 16:27 /project2/3240apr2013.dat
-rw-r--r--   1 hadoop supergroup    1251773 2019-10-23 16:27 /project2/3240aug2013.dat
-rw-r--r--   1 hadoop supergroup    1759808 2019-10-23 16:27 /project2/3240dec2013.dat
-rw-r--r--   1 hadoop supergroup    1603579 2019-10-23 16:27 /project2/3240feb2013.dat
-rw-r--r--   1 hadoop supergroup    1676139 2019-10-23 16:27 /project2/3240jan2013.dat
-rw-r--r--   1 hadoop supergroup    1549855 2019-10-23 16:27 /project2/3240jul2013.dat
-rw-r--r--   1 hadoop supergroup    1598095 2019-10-23 16:27 /project2/3240jun2013.dat
-rw-r--r--   1 hadoop supergroup    1604886 2019-10-23 16:27 /project2/3240mar2013.dat
-rw-r--r--   1 hadoop supergroup    1870898 2019-10-23 16:27 /project2/3240may2013.dat
-rw-r--r--   1 hadoop supergroup    1456666 2019-10-23 16:27 /project2/3240nov2013.dat
-rw-r--r--   1 hadoop supergroup    1461324 2019-10-23 16:27 /project2/3240oct2013.dat
-rw-r--r--   1 hadoop supergroup    1419624 2019-10-23 16:27 /project2/3240sep2013.dat
```

- **Step 2**: Loading data (only September data) into Pig and store as "chararray", using the following command:

  `lines = LOAD 'hdfs://localhost:8020/project2/3240sep2013.dat' AS (line:chararray);`

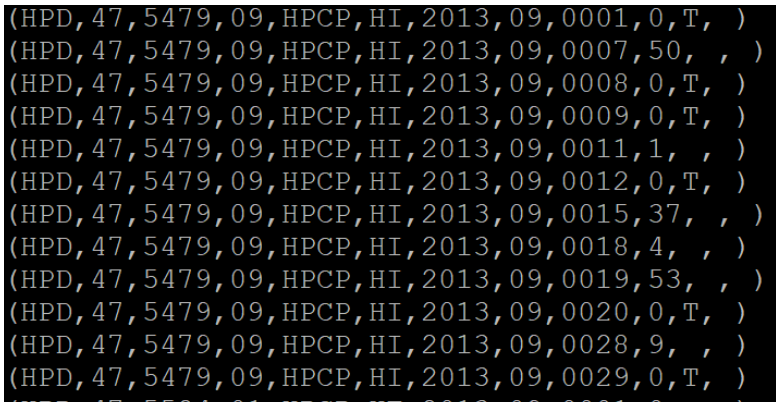
- **Step 3**: Chunking data fields into different column, separated by "tap" value according to Table 1, using the following command:

```
sept_data = FOREACH lines GENERATE SUBSTRING(line,0,3) AS
record_type,SUBSTRING(line,3,5) AS state,SUBSTRING(line,5,9) AS
co_network,SUBSTRING(line,9,11) AS co_division,
SUBSTRING(line,11,15) AS element_type,SUBSTRING(line,15,17) AS
element_unit, SUBSTRING(line,17,21) AS year,SUBSTRING(line,21,23)
AS month,SUBSTRING(line,23,27) AS day,(int)
SUBSTRING(line,(int)SIZE(line)-8,(int)SIZE(line)-2) as data_value
,SUBSTRING(line,(int)SIZE(line)-2,(int)SIZE(line)-1) AS flag1,
SUBSTRING(line,(int)SIZE(line)-1,(int)SIZE(line)) AS flag2;
```

- **Step 4**: Filtering data value, using following command**:**

```
filtered_sept_data = FILTER sept_data BY flag1 matches '[^I]';
```

`new tuples output:`



```
(HPD,47,5479,09,HPCP,HI,2013,09,0001,0,T, )
(HPD,47,5479,09,HPCP,HI,2013,09,0007,50,  , )
(HPD,47,5479,09,HPCP,HI,2013,09,0008,0,T, )
(HPD,47,5479,09,HPCP,HI,2013,09,0009,0,T, )
(HPD,47,5479,09,HPCP,HI,2013,09,0011,1,  , )
(HPD,47,5479,09,HPCP,HI,2013,09,0012,0,T, )
(HPD,47,5479,09,HPCP,HI,2013,09,0015,37,  , )
(HPD,47,5479,09,HPCP,HI,2013,09,0018,4,  , )
(HPD,47,5479,09,HPCP,HI,2013,09,0019,53,  , )
(HPD,47,5479,09,HPCP,HI,2013,09,0020,0,T, )
(HPD,47,5479,09,HPCP,HI,2013,09,0028,9,  , )
(HPD,47,5479,09,HPCP,HI,2013,09,0029,0,T, )
```

3) Using Queries to Find the Result

<u>3.1) On average, find the state that has highest precipitation in September 2013</u>

Using the new tuples from **filtered_sept_data** from step 4, section 2.

- **Step 1**: Group the filtered data by state, using the following command:

```
grouped_by_state = GROUP filtered_sept_data BY state;
```

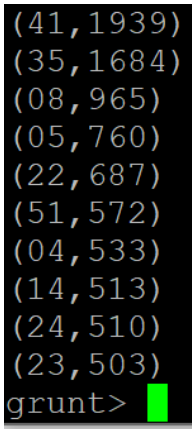
- **Step 2**: Find the average of each state, using the following command:

```
avg_of_state = FOREACH grouped_by_state GENERATE group as state,
 SUM(filtered_sept_data.data_value)/30 as avg_value;
```

- **Step 3**: Show the average order by descending and limit 10 results only, use the following commands:

```
order_by_avg = ORDER avg_of_state BY avg_value DESC;
max_avg_10 = LIMIT order_by_avg 10;
dump max_avg_10;
```

Therefore, the highest average precipitation in September 2013 goes with state ID 41, which is **Texas**, had **19.39** inches**.**



```
(41,1939)
(35,1684)
(08,965)
(05,760)
(22,687)
(51,572)
(04,533)
(14,513)
(24,510)
(23,503)
grunt>
```

3.2) find the state that has lowest precipitation in 2013

Using same process in section 2, except loading all files instead of only the “sept file”.

- **Step 1**: Loading all data in 2013 into Pig and store as “chararray”, using command:

```
lines = LOAD 'hdfs://localhost:8020/project2/*.dat' AS
(line:chararray);
```

- **Step 2**: Do the same as **Step 3** in section **2.**

```
all_year_data = FOREACH lines GENERATE SUBSTRING(line,0,3) AS
record_type,SUBSTRING(line,3,5) AS state,SUBSTRING(line,5,9) AS
co_network,SUBSTRING(line,9,11) AS
co_division,SUBSTRING(line,11,15) AS
element_type,SUBSTRING(line,15,17) AS
element_unit,SUBSTRING(line,17,21) AS year,SUBSTRING(line,21,23)
AS month,SUBSTRING(line,23,27) AS day,(double)
SUBSTRING(line,(int)SIZE(line)-8,(int)SIZE(line)-2) as
data_value;
```

- **Step 3**: Group the filtered data by state, using command:

```
grouped_by_state = GROUP all_year_data BY state;
```

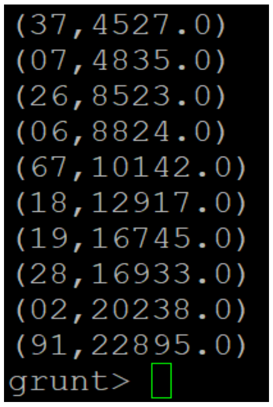
- **Step 5**: Sum all the precipitation data value by each state, using command:

```
sum_of_state = FOREACH grouped_by_state GENERATE group as state,
SUM(all_year_data.data_value) as sum_value;
```

- **Step 6**: Show the sum order by ascending and limit 10 results only, use commands:

```
order_by_sum_of_state = ORDER sum_of_state BY sum_value ASC;
lowest_sum_of_state_10 = LIMIT order_by_sum_of_state 10;
dump lowest_sum_of_state_10;
```

Therefore, the lowest precipitation in 2013 goes with state ID 37, which is **Rhode Island**, had only **45.27** inches.



```
(37,4527.0)
(07,4835.0)
(26,8523.0)
(06,8824.0)
(67,10142.0)
(18,12917.0)
(19,16745.0)
(28,16933.0)
(02,20238.0)
(91,22895.0)
grunt>
```

3.3) Find the state had more than an inch of precipitation on July 4, 2013

Using same process in section 2, by loading only the “july file”.

- **Step 1**: Loading data (only July) into Pig and store as “chararray”, using command:

```
lines = LOAD 'hdfs://localhost:8020/project2/3240jul2013.dat’ AS

(line:chararray);
```

- **Step 2**: Chunk data fields into different column, using the following command:

```
jul_data = FOREACH lines GENERATE SUBSTRING(line,0,3) AS
record_type,SUBSTRING(line,3,5) AS state,SUBSTRING(line,5,9) AS
co_network,SUBSTRING(line,9,11) AS co_division,SUBSTRING(line,11,15)
AS element_type,SUBSTRING(line,15,17) AS
element_unit,SUBSTRING(line,17,21) AS year,SUBSTRING(line,21,23) AS
month,SUBSTRING(line,23,27) AS day,(double)
SUBSTRING(line,(int)SIZE(line)-8,(int)SIZE(line)-2) as data_value;
```

- **Step 3**: Filter out by keeping only the day is 4th, using the following command:

```
july_4_data = FILTER jul_data BY day == '0004';
```

- **Step 4**: Group the filtered data by state, using command:

```
grouped_by_state = GROUP july_4_data BY state;
```

- **Step 5**: Sum all the precipitation data value by each state, using command:

```
sum_of_state = FOREACH grouped_by_state GENERATE group as state,
SUM(filtered_jul_data.data_value)/100 as sum_value;
```

*** *Note: the value is divided by 100 because the original data value is scaled 1/100 (inch)*

Example: if the data value is 00030, the precipitation value is 0.3 inches.

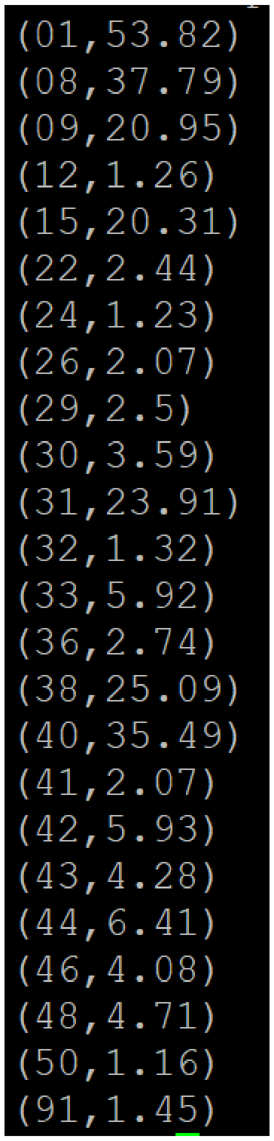
- **Step 6**: Show the states that had more than 1 inch, using the following command:

```
states_has_more_inch_pres = FILTER sum_of_state BY sum_value > 1;
dump states_has_more_inch_pres;
```

`

Therefore, the states that had precipitation more than 1 inch are:

```
state ID:01 is Alabama, 53.82 inches
state ID:08 is Delaware, 37.79 inches
state ID:09 is Georgia, 20.95 inches
state ID:15 is Kentucky, 20.31 inches
state ID:22 is Mississippi, 2.44 inches
state ID:24 is Montana, 1.23 inches
state ID:26 is Nevada, 2.07 inches
state ID:29 is New Mexico, 2.5 inches
state ID:30 is New York, 3.59 inches
state ID:31 is North Carolina, 23.91 inches
state ID:32 is North Dakota, 1.32 inches
state ID:33 is Ohio, 5.92 inches
state ID:36 is Pennsylvania, 2.74 inches
state ID:38 is South Carolina, 25.09 inches
state ID:40 is Tennessee, 35.49 inches
state ID:41 is Texas, 2.07 inches
state ID:42 is Utah, 5.93 inches
state ID:43 is Vermont, 4.28 inches
state ID:44 is Virginia, 6.41 inches
state ID:46 is West Virginia, 4.08 inches
state ID:48 is Wyoming, 4.71 inches
state ID:50 is Alaska, 1.16 inches
state ID:91 is Pacific Islands, 1.45 inches
```



```
(01,53.82)
(08,37.79)
(09,20.95)
(12,1.26)
(15,20.31)
(22,2.44)
(24,1.23)
(26,2.07)
(29,2.5)
(30,3.59)
(31,23.91)
(32,1.32)
(33,5.92)
(36,2.74)
(38,25.09)
(40,35.49)
(41,2.07)
(42,5.93)
(43,4.28)
(44,6.41)
(46,4.08)
(48,4.71)
(50,1.16)
(91,1.45)
```

Appendix - Command List

Solution 3.1

-
  ```
  $sudo hadoop fs -put '/home/bitnami/project2/' '/project2'
  ```

-
  ```
  $sudo hadoop fs -ls /project2
  ```

-
  ```
  lines = LOAD 'hdfs://localhost:8020/project2/ 3240sep2013.dat’ AS
  (line:chararray);
  ```

-
  ```
  sept_data = FOREACH lines GENERATE SUBSTRING(line,0,3) AS
  record_type,SUBSTRING(line,3,5) AS state,SUBSTRING(line,5,9) AS
  co_network,SUBSTRING(line,9,11) AS co_division,
  SUBSTRING(line,11,15) AS element_type,SUBSTRING(line,15,17) AS
  element_unit, SUBSTRING(line,17,21) AS year,SUBSTRING(line,21,23) AS
  month,SUBSTRING(line,23,27) AS day,(int)
  SUBSTRING(line,(int)SIZE(line)-8,(int)SIZE(line)-2) as data_value
  ,SUBSTRING(line,(int)SIZE(line)-2,(int)SIZE(line)-1) AS flag1,
  SUBSTRING(line,(int)SIZE(line)-1,(int)SIZE(line)) AS flag2;
  ```

-
  ```
  filtered_sept_data = FILTER sept_data BY flag1 matches '[^I]';
  ```

-
  ```
  grouped_by_state = GROUP filtered_sept_data BY state;
  ```

-
  ```
  avg_of_state = FOREACH grouped_by_state GENERATE group as state,
   SUM(filtered_sept_data.data_value)/30 as avg_value;
  ```

-
  ```
  order_by_avg = ORDER avg_of_state BY avg_value DESC;
  max_avg_10 = LIMIT order_by_avg 10;
  ```

-
  ```
  dump max_avg_10;
  ```

Solution 3.2

-
  ```
  lines = LOAD 'hdfs://localhost:8020/project2/*.dat' AS
  (line:chararray);
  ```

-
  ```
  all_year_data = FOREACH lines GENERATE SUBSTRING(line,0,3) AS
  record_type,SUBSTRING(line,3,5) AS state,SUBSTRING(line,5,9) AS
  co_network,SUBSTRING(line,9,11) AS co_division,SUBSTRING(line,11,15)
  AS element_type,SUBSTRING(line,15,17) AS
  element_unit,SUBSTRING(line,17,21) AS year,SUBSTRING(line,21,23) AS
  month,SUBSTRING(line,23,27) AS day,(double)
  SUBSTRING(line,(int)SIZE(line)-8,(int)SIZE(line)-2) as data_value;
  ```

-
  ```
  grouped_by_state = GROUP all_year_data BY state;
  ```

-
  ```
  sum_of_state = FOREACH grouped_by_state GENERATE group as state,
  SUM(all_year_data.data_value) as sum_value;
  ```

-
  ```
  order_by_sum_of_state = ORDER sum_of_state BY sum_value ASC;
  ```

-
  ```
  lowest_sum_of_state_10 = LIMIT order_by_sum_of_state 10;
  ```

-
  ```
  dump lowest_sum_of_state_10;
  ```

Solution 3.3

-
  ```
  lines = LOAD 'hdfs://localhost:8020/project2/ 3240sep2013.dat’ AS
  (line:chararray);
  ```

-
  ```
  jul_data = FOREACH lines GENERATE SUBSTRING(line,0,3) AS
  record_type,SUBSTRING(line,3,5) AS state,SUBSTRING(line,5,9) AS
  co_network,SUBSTRING(line,9,11) AS co_division,SUBSTRING(line,11,15)
  AS element_type,SUBSTRING(line,15,17) AS
  element_unit,SUBSTRING(line,17,21) AS year,SUBSTRING(line,21,23) AS
  month,SUBSTRING(line,23,27) AS day,(double)
  SUBSTRING(line,(int)SIZE(line)-8,(int)SIZE(line)-2) as data_value;
  ```

-
  ```
  july_4_data = FILTER jul_data BY day == '0004';
  ```

-
  ```
  grouped_by_state = GROUP july_4_data BY state;
  ```

-
  ```
  sum_of_state = FOREACH grouped_by_state GENERATE group as state,
  SUM(filtered_all_year_data.data_value) as sum_value;
  ```

-
  ```
  order_by_sum_of_state = ORDER sum_of_state BY sum_value ASC;
  lowest_sum_of_state_10 = LIMIT order_by_sum_of_state 10;
  ```

-
  ```
  dump lowest_sum_of_state_10;
  ```

References

Data Description

ftp://ftp.ncdc.noaa.gov/pub/data/hourly_precip-3240/dsi3240.pdf

Data of Each Month in 2013

ftp://ftp.ncdc.noaa.gov/pub/data/hourly_precip-3240/by_month2013/

Data Provider

https://www.data.gov/