Analysis on H1-B Visa Petitions using Big Data Technologies

CWID	Group Member Name
A20430008	SIDDHI SANJAY KULKARNI
A20429199	SHIVANGEE DURGADAS KULKARNI

CSP 554 Big Data Technologies

Professor: Adam McElhinney

November 25th, 2019

Introduction:

Now-a-days there is a variety of the information available from different websites. It is becoming very difficult to handle this data and so we need to process this data. The traditional data processing techniques are not capable of handling this huge amount of data and so there is a need for the Big Data technologies to handle and process this data. The tools like Hadoop, Map Reduce, etc. and other tools are capable of handling this data. [1] In this project we have selected the H1-B data of the employees and are going to analyze this data. We have used some of the test cases for analyzing the data in Hive, Pig, Spark, and Map Reduce. Also, we will compare the query execution time for the queries executed in Hive, Pig and Spark SQL.

Map Reduce:

Map Reduce – Map reduce is a software framework for processing of data in a distributed fashion over different machine. The core idea behind map reduce is mapping your data set into a collection of <key, value> pairs and then reducing overall pairs with the same key.[2]

Map Reduce program works in two phases:

1. Map phase (Splits and Mapping)
2. Reduce phase (Shuffling and Reducing)

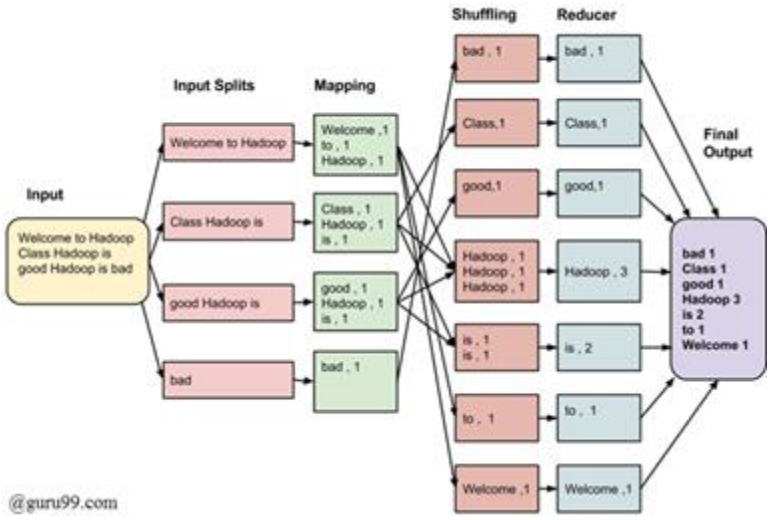


Fig.1: Architecture Of Map Reduce [2]

The data goes through the following phases:

Input Split: An input is split into fixed sized pieces called input split. It is the chunk of the input.

Mapping: This is the first step of the execution of a map-reduce program. The data is passed to the mapper function to generate the key,value pairs.

Shuffling: This phase takes the input of the mapping phase and consolidates the relevant records from the map phase.

Reducing: The output from the shuffling phase are aggregated.

The complete Map Reduce architecture is controlled by two entites:

1. Job Tracker- Acts like a master.
2. Multiple Task Tracker- Acts like a slave.

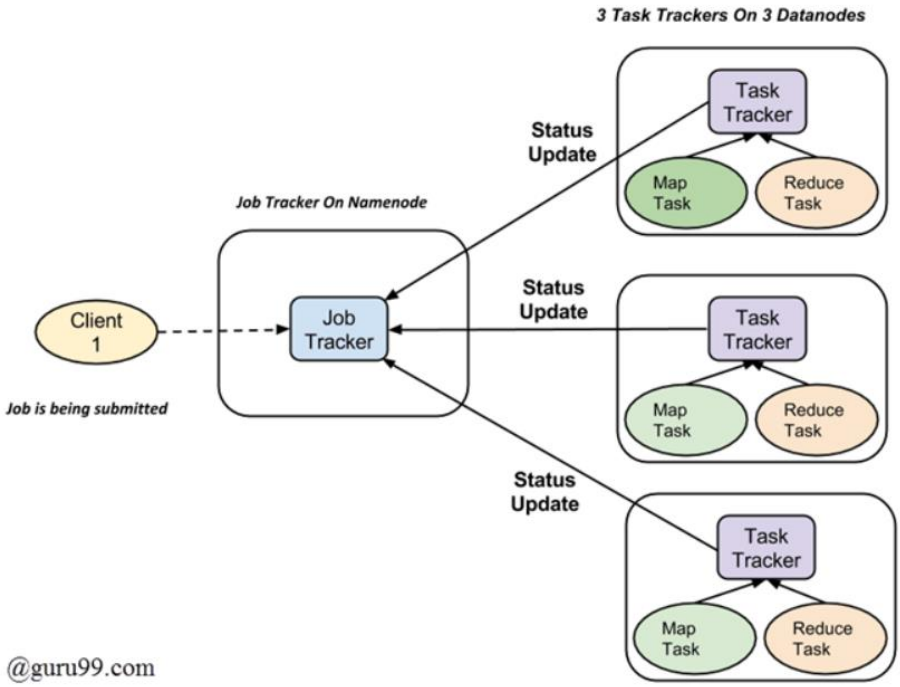


Fig.2: Job Execution in Map Reduce

The Job tracker resides on the Name node and the Task Tracker resides on the Data Nodes.

1. A job is divided into multiple tasks which are then run onto multiple data nodes in a cluster.
2. The job tracker is responsible to coordinate the activity by scheduling the tasks to run on different data nodes.
3. Execution of individual task is then looked by the task tracker.
4. Task tracker sends the heartbeat signal to the job tracker and the task completion report to the Job Tracker to notify him the current state of the system.[2]

Advantages:

1. Hadoop is a platform that is highly scalable. It is highly scalable.
2. MapReduce supports parallel programming. It divides tasks and executes parallely.

Disadvantages:

1. Mapreduce cannot be used in Real-time processing.

2. Mapreduce cannot be used in OLTP.

Apache Hive:

Apache Hive is an open source data warehouse system built on top of Hadoop Housed for querying and analyzing large datasets stored in Hadoop files. It processes structured and semi-structured data in Hadoop. Hive requires one additional component known as Metastore which is not present in Hadoop. It stores metadata for each of the tables like their schema and location. Hive also includes the partition metadata. [3]

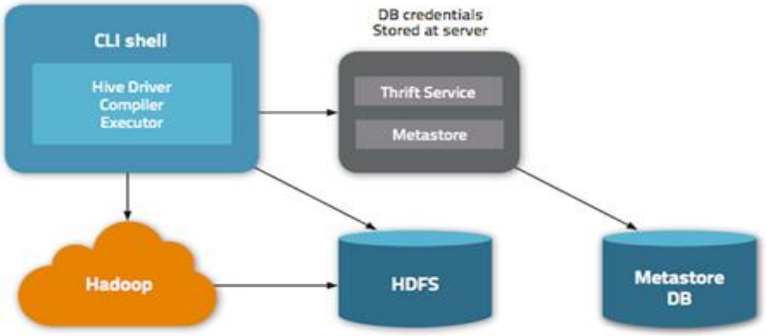


Fig.3: Architecture Of Hive

The architecture of Hive includes the following components:

1. Metastore- It stores the metadata for each of the tables like schema and location. It also includes partition metadata.
2. Driver- It acts like a controller for the execution of the HiveQL statements by creating sessions. It monitors the lifecycle and the progress of the query execution.
3. Compiler- It compiles the Hive query. It converts the query to an execution plan. The compiler converts the query to an Abstract Syntax Tree (AST). Then the AST is converted to a Directed Acyclic Graph (DAG).
4. Optimizer- It performs the various transformations on the execution plan to provide optimized DAG. The optimizer can perform the operations like splitting the tasks and then applying the transformation on data before performing the reduce operation.
5. Executor- It is responsible for the execution of tasks.[3]

Advantages:

1. Hive helps querying larger datasets residing in distributed storage.
2. Multiple users can simultaneously query the data using Hive-QL.
3. Data extract/transform/load can be done easily.

Disadvantages:

1. It is not designed for Online Transaction Processing (OLTP), it is only used for Online Analytical Processing (OLAP).
2. Hive does not support Sub-queries.

Apache Pig:

Pig is a high-level programming language useful for analyzing large data sets. A pig was a result of development effort at Yahoo! It enables focus on analyzing the bulk data sets and reduce the time for the execution of the Map Reduce programs.

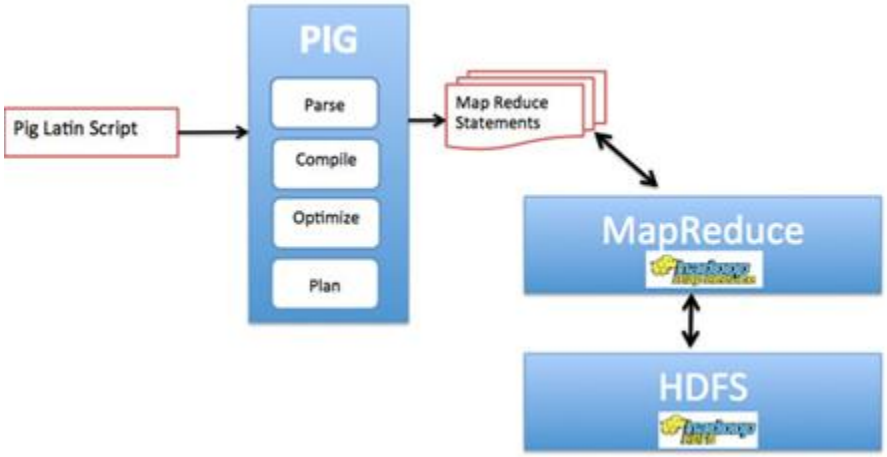


Fig.4: Architecture Of Pig

The components of the Pig architecture are as follows:[5]

Parser- It performs the syntax check of the script and the other miscellaneous checks. The output of the parser will be a DAG.

Optimizer- The DAG is passed on to the logical optimizer for carrying out the logical operations like projection and pushdown.

Compiler- It compiles the logical plan into a series of MapReduce jobs.

Execution Engine- The map reduce jobs are submitted to Hadoop in a sorted order. The jobs are executed on Hadoop producing the desired results.[4]

Advantages:

1. It is a dataflow structure.
2. Pig is effective for unstructured.
3. Pig uses lazy evaluation.

Disadvantages:

1. Pig delays execution. The final result is displayed only after final command is executed.
2. In Apache pig, data schema is enforced implicitly. This is the disadvantage of pig.

Apache Spark:

Apache Spark is an open source, Hadoop-compatible, fast and general-purpose cluster computing platform. It is a powerful open-source engine that provides real-time stream processing, interactive processing, graph processing, in-memory processing as well as batch processing with very fast speed, ease of use and standard interface.[5]

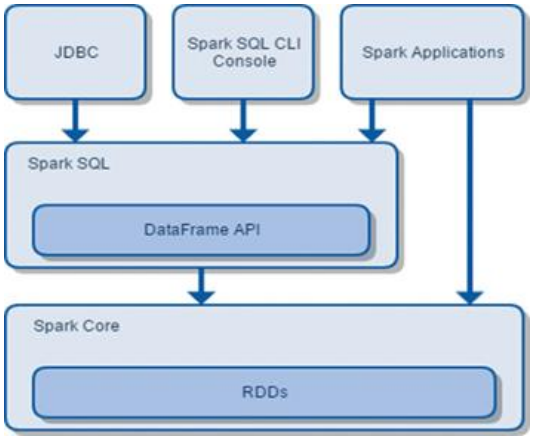


Fig.5: Architecture Of Apache Spark

The architecture of Spark includes the following components:

1. Spark Core- It is a kernel of the Spark which provides an execution platform for all the Spark applications. Resilient Distributed Dataset (RDD) is a fundamental unit of Apache Spark which is a collection of elements across cluster nodes that can perform parallel operations. The RDDs are immutable but can generate a new RDD by transforming existing RDD. There are three ways to create RDD in Spark:

   1. Parallel Collections- We can create this RDD by invoking the parallelize method in driver program.

   2. External datasets- One can create RDD by calling the textFile method. This method takes the URL of a file and reads it as a collection of lines.

   3. Existing RDDs- By applying transformation on the existing RDD.

2. Spark SQL- It enables the users to run the SQL/HQL queries on the Spark. It allows us to process the structured as well as unstructured data.

Advantages:

1. It is easy to use also speed of SPARK is better than other tools like Apache hive and pig.
2. Advanced analytics is used in spark.
3. Apache spark is dynamic in nature.

Disadvantages:

1. There is no automatic optimization process in Apache Spark.
2. Apache spark is not a suit for a multiuser environment.

Implementation:

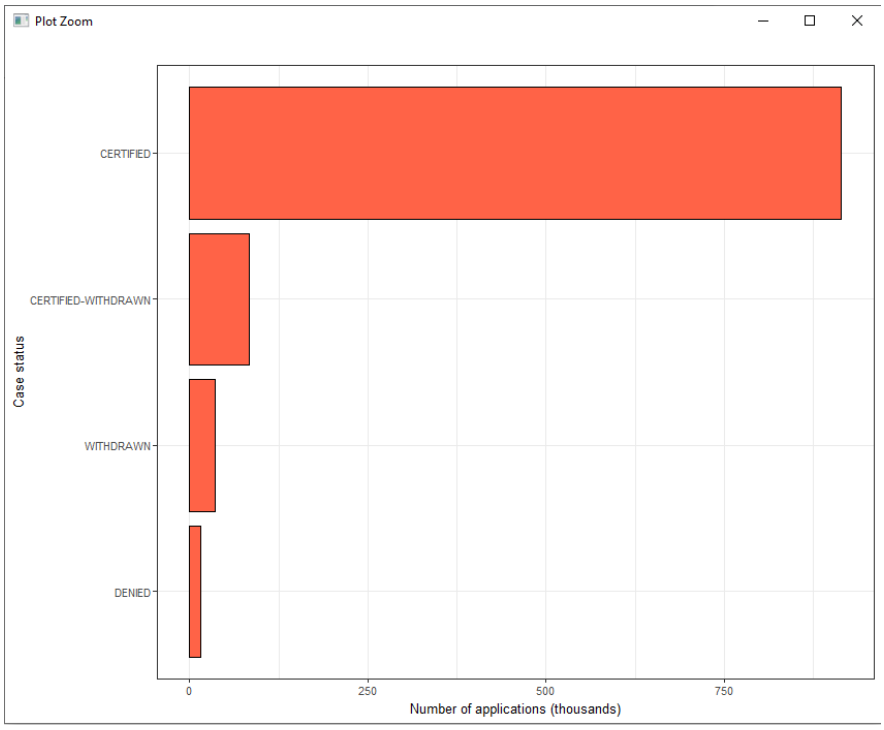
Step1: Understanding dataset [6]:

- **The meaning of the columns are as follows**:

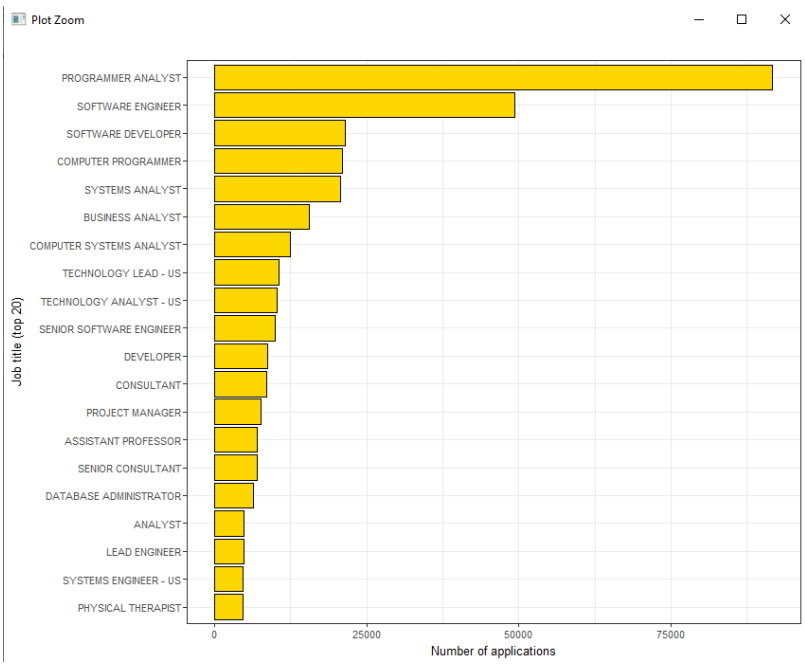
- Case_status – Status of the application
- Employer_name – Name of employer registered in H1B visa application
- Soc_name – Occupation code for employment
- Job_title – Job title for employment
- Full_time_position – Whether application is for full-time position or part-time position
- Prevailing_wage – The most frequent wage
- Year – The application year
- Worksite – The address of the employer worksite

➢ **Data Visualization**:

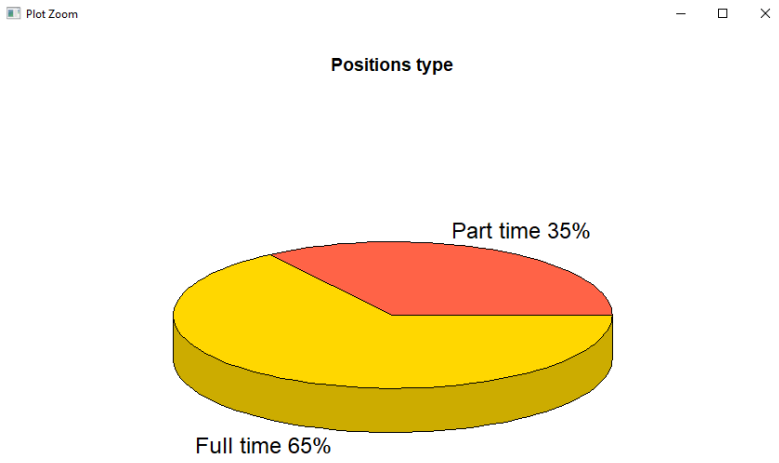
1. Distribution of CASE_STATUS:



2. Top 20 of most frequent job titles in visa applications:

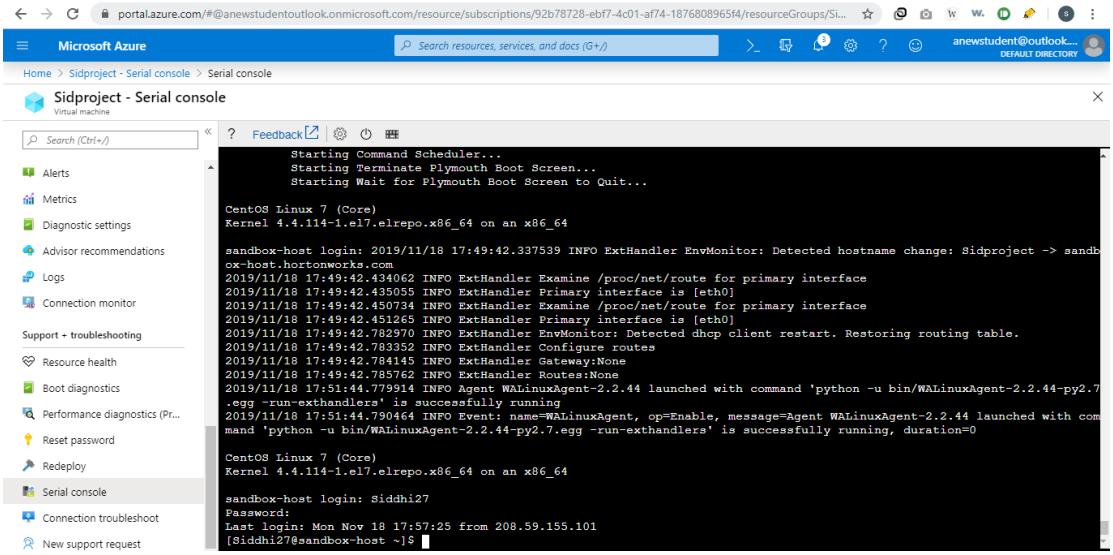


3. Percent of full-time positions from the total number of applications:



Step 2: Step by step execution

1. **Starting the virtual machine:**



2. **<u>Starting with Hive:</u>**

Load the data in Hive

```
hive> load data local inpath 'h1b_data.csv' into table big_data.h1b_data_new;
Loading data to table big_data.h1b_data_new
Table big_data.h1b_data_new stats: [numFiles=1, numRows=0, totalSize=492258374, rawDataSize=0]
OK
Time taken: 2.329 seconds
hive>
```

Is number of petitions with Data Engineer job title increasing over time?

```
hive> select year, count(year) from big_data.h1b_data_new
    > where job_title like '%DATA ENGINEER%'
    > group by year;
Query ID = maria_dev_20191120031838_372a1d96-ad9b-4b94-be65-fc9e67134b67
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1574212744954_0005)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED     13         13        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 45.11 s
--------------------------------------------------------------------------------
OK
2011    60
2012    81
2013    151
2014    249
2015    394
2016    786
Time taken: 45.858 seconds, Fetched: 6 row(s)
hive>
```

3. Starting with Pig:

Load data in Pig

```
grunt> DEFINE CSVLoader org.apache.pig.piggybank.storage.CSVLoader();
grunt> h1b_data = LOAD '/user/big_data_project/dataset.csv' using CSVLoader(',') AS (sr_no:int, case_status:chararray, Employer_name:chararray, Soc_name:chararray, Job_
title:chararray, full_time_position:chararray, Prevailing_wage:chararray, year:chararray, worksite:chararray, longitude:double, latitude:double);
grunt> limit_data = limit h1b_data 10;
grunt> dump limit_data;
2019-11-20 17:49:15,551 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: LIMIT
2019-11-20 17:49:15,613 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2019-11-20 17:49:15,661 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, ConstantCalculator,
GroupByConstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, PartitionFilterOptimizer, PredicatePushdownOptimizer, PushDownForEachFlatte
n, PushUpFilter, SplitFilter, StreamTypeCastInserter]}
2019-11-20 17:49:15,786 [main] INFO  org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (PS Old Gen) of size 699400192 to monitor. collectionUsageThreshold
 = 489580128, usageThreshold = 489580128
2019-11-20 17:49:15,905 [main] INFO  org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter - File Output Committer Algorithm version is 1
2019-11-20 17:49:15,905 [main] INFO  org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter - FileOutputCommitter skip cleanup _temporary folders under output direc
tory:false, ignore cleanup failures: false
2019-11-20 17:49:15,998 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2019-11-20 17:49:16,104 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2019-11-20 17:49:16,124 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2019-11-20 17:49:16,131 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
2019-11-20 17:49:16,501 [main] INFO  org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter - Saved output of task 'attempt__0001_m_000001_1' to hdfs://sandbox-hdp.
hortonworks.com:8020/tmp/temp1046299418/tmp-1893497295/_temporary/0/task__0001_m_000001
2019-11-20 17:49:16,562 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2019-11-20 17:49:16,573 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2019-11-20 17:49:16,574 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(,CASE_STATUS,EMPLOYER_NAME,SOC_NAME,JOB_TITLE,FULL_TIME_POSITION,PREVAILING_WAGE,YEAR,WORKSITE,,)
(1,CERTIFIED-WITHDRAWN,UNIVERSITY OF MICHIGAN,BIOCHEMISTS AND BIOPHYSICISTS,POSTDOCTORAL RESEARCH FELLOW,N,36067,2016,ANN ARBOR, MICHIGAN,-83.7430378,42.2808256)
(2,CERTIFIED-WITHDRAWN,GOODMAN NETWORKS, INC.,CHIEF EXECUTIVES,CHIEF OPERATING OFFICER,Y,242674,2016,PLANO, TEXAS,-96.6988856,33.0198431)
(3,CERTIFIED-WITHDRAWN,PORTS AMERICA GROUP, INC.,CHIEF EXECUTIVES,CHIEF PROCESS OFFICER,Y,193066,2016,JERSEY CITY, NEW JERSEY,-74.0776417,40.7281575)
(4,CERTIFIED-WITHDRAWN,GATES CORPORATION, A WHOLLY-OWNED SUBSIDIARY OF TOMKINS PLC,CHIEF EXECUTIVES,REGIONAL PRESIDEN, AMERICAS,Y,220314,2016,DENVER, COLORADO,-104.9902
51,39.7392358)
(5,WITHDRAWN,PEABODY INVESTMENTS CORP.,CHIEF EXECUTIVES,PRESIDENT MONGOLIA AND INDIA,Y,157518.4,2016,ST. LOUIS, MISSOURI,-90.1994042,38.6270025)
(6,CERTIFIED-WITHDRAWN,BURGER KING CORPORATION,CHIEF EXECUTIVES,EXECUTIVE V P, GLOBAL DEVELOPMENT AND PRESIDENT, LATIN AMERI,Y,225000,2016,MIAMI, FLORIDA,-80.1917902,25
.7616798)
(7,CERTIFIED-WITHDRAWN,BT AND MK ENERGY AND COMMODITIES,CHIEF EXECUTIVES,CHIEF OPERATING OFFICER,Y,91021,2016,HOUSTON, TEXAS,-95.3698028,29.7604267)
(8,CERTIFIED-WITHDRAWN,GLOBO MOBILE TECHNOLOGIES, INC.,CHIEF EXECUTIVES,CHIEF OPERATIONS OFFICER,Y,150000,2016,SAN JOSE, CALIFORNIA,-121.8863286,37.3382082)
(9,CERTIFIED-WITHDRAWN,ESI COMPANIES INC.,CHIEF EXECUTIVES,PRESIDENT,Y,127546,2016,MEMPHIS, TEXAS,,)
grunt>
```

Which worksite has popular job title?

```
grunt> Limitdata = foreach h1b_data generate worksite, Job_title;
grunt> group1 = group Limitdata by (worksite, Job_title);
grunt> count_job = foreach group1 generate FLATTEN(group) as (worksite, Job_title), COUNT(Limitdata) as count_no;
grunt> after = group count_job by Job_title;
grunt> final = foreach after { sort = order count_job by Job_title desc; limiter = LIMIT sort 10; generate limiter;};
grunt> query1 = foreach final generate FLATTEN(limiter);
grunt>
```

4. **Starting with Spark:**

Load data in Spark SQL

```
>>> df = sqlContext.read.load('/user/big_data_project/dataset.csv', format='com.databricks.spark.csv', header='true', inferSchema='true')
>>> df.count()
3002458
```

Is number of petitions with Data Engineer job title increasing over time?

```
>>> query2 = spark.sql("select year, count(year) from h1b where Job_title like '%DATA ENGINEER%' group by year").show()
+----+-----------+
|year|count(year)|
+----+-----------+
|2016|        786|
|2012|         81|
|2014|        249|
|2013|        151|
|2011|         60|
|2015|        394|
+----+-----------+

>>>
```

Experimentation with Hive, Pig and Spark processing environment:

The amount of time consumed during input a user query for finding records from the Hive technique is termed here as the query execution time. In order to measure the query execution time, below listed queries are fired on the Hive interphase and their performance is observed. After completing the observations first time for all the queries.

Queries:

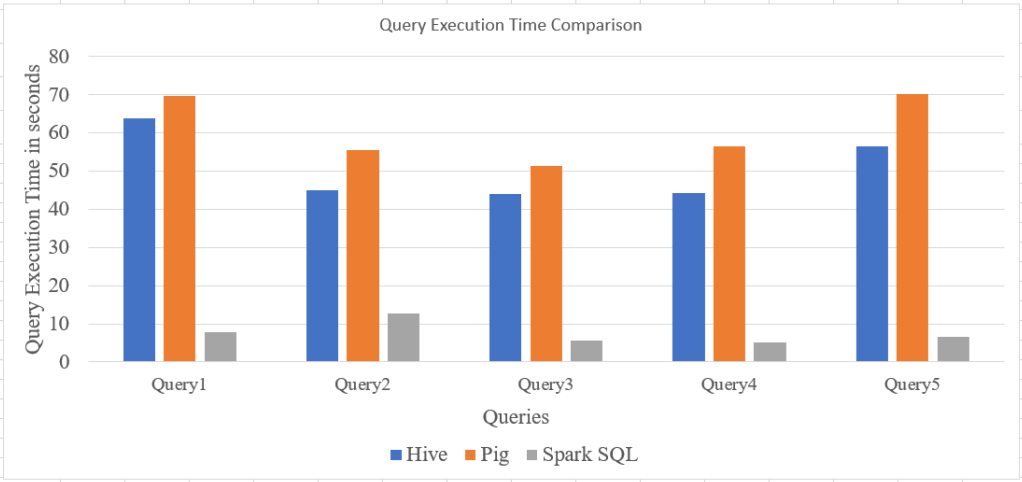
1) Which worksite has the popular job title?
2) Is the number of petitions with Data Engineer job title increasing over time?
3) Which part of US has most Data Engineer jobs?
4) Which industry has the greatest number of Data Scientist positions?
5) Which employers file the most petitions each year?
6) What is the general trend of the employer's H1-B status in the period 2011-16?
7) What is the general trend of the employer name with respect to the year?

Results and Discussion:

Compare performance of Hive, Pig and Spark

From the execution of the queries mentioned above we conclude that Spark is more efficient than Hive and Pig. This is so because Spark uses less time for execution of queries, less lines of code and the structure of the output is easy to understand than Hive and Pig. The screenshot below displays the query execution time of each tool and its pictorial view of the comparison.

	Hive	Pig	Spark SQL
Query1	63.73	69.7	7.88
Query2	45.11	55.63	12.72
Query3	44.13	51.41	5.71
Query4	44.32	56.45	5.06
Query5	56.44	70.14	6.67



Conclusion:

Thus, the results represent that the performance of the Spark SQL is better than Hive and Pig. Spark is promising of enhanced performance which is 100 times faster than Hadoop MapReduce for some applications. Spark also supports the low-latency in-memory data processing capability which is missing in the Map Reduce.

References:

[1]. _Monu, Monika and Pal, Sat, Simulation of Performance Analysis of MongoDB, PIG, HIVE Storage, Map Reduce, Spark and Yarn (April 4, 2019). Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur - India, February 26-28, 2019. Available at SSRN: https://ssrn.com/abstract=3365403 or http://dx.doi.org/10.2139/ssrn.3365403

[2] https://www.guru99.com/introduction-to-mapreduce.html

[3] https://data-flair.training/blogs/apache-hive-tutorial/

[4] https://www.tutorialspoint.com/apache_pig/apache_pig_architecture.htm

[5] https://data-flair.training/blogs/what-is-spark/

Dataset:

[6] H1B dataset (2011-16). Retrieved from https://www.kaggle.com/nsharan/h-1b-visa

Appendix:

a) Data understanding and data visualization Code in R:

```
library(knitr)
library(kableExtra)
library(dplyr)
library(ggplot2)
library(plotrix)

df = read.csv("h1b.csv", header = TRUE, sep = ",")

#group the data on CASE_STATUS to see the distribution of CASE_STATUS.
df %>% filter(!is.na(CASE_STATUS)) %>% group_by(CASE_STATUS) %>%
summarise(nr = length(lat)) %>% ungroup() -> dc
ggplot(data = dc, aes(x = reorder(CASE_STATUS,nr), y = nr/1000)) +
geom_bar(stat="identity", fill="tomato", colour="black") +
coord_flip() + theme_bw(base_size = 10)  +
labs(title="", x ="Case status", y = "Number of applications (thousands)")


#top 20 of most frequent job titles in visa applications
df %>% group_by(JOB_TITLE) %>% summarise(nr = length(lat)) %>%
top_n(n=20) %>% arrange(-nr) %>% ungroup() -> dj
ggplot(data = dj, aes(x = reorder(JOB_TITLE,nr), y = nr)) +
geom_bar(stat="identity", fill="gold", colour="black") +
coord_flip() + theme_bw(base_size = 10)  +
labs(title="", x ="Job title (top 20)", y = "Number of applications")



#percent of full-time positions from the total number of applications
df %>% filter(!is.na(FULL_TIME_POSITION)) %>% group_by(FULL_TIME_POSITION) %>% summarise(nr = length(lat)) %>% ungroup() -> dp
lbls = c("Part time","Full time")
pcts = round(dp$nr / sum(dp$nr) * 100,0)
lbls = paste(lbls, pcts)
lbls = paste(lbls,"%", sep="")
cols = c("tomato", "gold")
pie3D(x=dp$nr, labels=lbls, col = cols, explode=0, main = "Positions type")
```

b) Hive commands:

```
# HIVE COMMANDS

load data local inpath 'h1b_data.csv' into table big_data.h1b_data_new;

#Query 1:Which worksite has popular job title?
select worksite, job_title, count(job_title) from h1b_data_new
group by worksite, job_title
Order by job_title desc;
limit 50;

#Query 2: Is number of petitions with Data Engineer job title increasing over time?
select year, count(year) from big_data.h1b_data_new
where job_title like '%DATA ENGINEER%'
Group by year;

#Query 3: Which part of US has most data engineer jobs?
select worksite, count(worksite) from h1b_data_new
where job_title like '%DATA ENGINEER%
Group by worksite
Order by worksite
limit 20;

#Query 4: Which industry has greatest number of Data scientist position?
select soc_name, Count(soc_name) from h1b_data_new
where job_title like '%DATA SCIENTIST%'
Group by soc_name
Order by soc_name
limit 20;


#query 5: Which employer files the most petitions each year?
select employer_name, year, count(year) from h1b_data_new
group by year employer_name
order by year
limit 20;
```

```
#Query 6: What is the general trend of the employerâ€™s H1b status in the period 2011-16?
select year, count(year) from h1b_data_new
where case_status = 'CERTIFIED'
group by year
Order by year desc;

#Query 7: What is general trend of the employer name with respect to the year?
select year, employer_name, count(employer_name) from h1b_data_new
group by employer_name, year
order by employer_name
limit 20;
```

c) Pig commands:

```
#Loading csv file
>DEFINE CSVLoader org.apache.pig.piggybank.storage.CSVLoader();
>h1b_data = LOAD '/user/big_data_project/dataset.csv' using CSVLoader(',') AS (Sr_no:int, case_status:chararray, Employer_name:chararray, Soc_name:chararray,
 Job_title:chararray, full_time_position:chararray, prevailing_wage:chararray, year:chararray, worksite:chararray, longitude:double, latitude:double);
>limit_data = LIMIT h1b_data 10;
>dump limit_data;

# Query 1 : Which worksite has popular job title?
>Limitdata = foreach h1b_data generate worksite, Job_title;
>group1 = group Limitdata by (worksite, Job_title);
>count_job = foreach group1 generate FLATTEN(group) as (worksite, Job_title), COUNT(Limitdata) as count_no;
>after = group count_job by job_title;
>final = foreach after { sort = order count_job by Job_title desc; limiter = LIMIT sort 10; Generate limiter;};
>query1 = foreach final generate FLATTEN(limiter);

# Query 2 : Is number of petitions with Data Engineer job title increasing over time?
>Limit_data_1 = foreach h1b_data generate year, Job_title;
>filter_job = FILTER Limit_data_1 by job_title matches 'DATA ENGINEER';
>group_yr = Group filter_job by year;
>count_yr = foreach group_yr generate filter_job.year, COUNT(filter_job);
>lim_count_yr = Limit count_yr 10;
>dump lim_count_yr;

# Query 3 : Which part of US has most data engineer jobs?
>filter_data = filter h1b_data by Job_title matches 'DATA ENGINEER';
>lim_data = foreach filter_data generate worksite;
>group_work = group lim_data by worksite;
>count_work = foreach group_work generate lim_data.worksite, COUNT(limit_data);
>lim_count = LIMIT count_work 10;
>dump lim_count;

#Query 4 : Which industry has greatest number of Data scientist position?
>lim_d = foreach h1b_data by Soc_name, Job_title;
>filter_d = FILTER lim_d by Job_title matches 'DATA SCIENTIST';
>group_d = Group filter_d by Soc_name;
>count_d = foreach group_d generate filter_d.Soc_name, COUNT(filter_d);
>limit_count_d = LIMIT count_d 10;
>dump limit_count_d;
```

```
#Query 5 : Which employer files the most petitions each year?
>liml = foreach h1b_data generate Employer_name, year;
>group1 = group liml by (Employer_name,year);
>count_yr = Foreach group1 generate liml.Employer_name, liml.year, COUNT(liml);
>lim_count_yr = LIMIT count_yr 10;
>dump lim_count_yr;
```

d) Spark SQL commands:

```
# SPARK COMMANDS

>df = sqlContext.read.load('/user/big_data_project/dataset.csv', format = 'com.databricks.spark.csv', header = 'true', inferSchema='true')
>df.count()
>df.dtypes

# Creating new temporary view
>df.createOrReplaceTempView("h1b")

#Query 1:Which worksite has popular job title?
>query1 = spark.sql("select worksite, job_title, count(job_title) from h1b group by worksite, job_title Order by job_title desc").show()


#Query 2: Is number of petitions with Data Engineer job title increasing over time?
>query2 = spark.sql("select year, count(year) from big_data.h1b where job_title like '%DATA ENGINEER%' Group by year").show()

#Query 3: Which part of US has most data engineer jobs?
>query3 = spark.sql("select worksite, count(worksite) from h1b where job_title like '%DATA ENGINEER%' Group by worksite Order by worksite").show()


#Query 4: Which industry has greatest number of Data scientist position?
>query4 = spark.sql("select soc_name, Count(soc_name) from h1b where job_title like '%DATA SCIENTIST%' Group by soc_name Order by soc_name").show()


#query 5: Which employer files the most petitions each year?
>query5 = spark.sql("select employer_name, year, count(year) from h1b group by year,employer_name order by year").show()

#Query 6: What is the general trend of the employerâ€™s H1b status in the period 2011-16?
>query6 = spark.sql("select year, count(year) from h1b where case_status = 'CERTIFIED' group by year Order by year desc").show()

#Query 7: What is general trend of the employer name with respect to the year?
>query7 = spark.sql("select year, employer_name, count(employer_name) from h1b group by employer_name, year order by employer_name").show()
```

e) Map Reduce:

```
from mrjob.job import MRJob
class ProjectMapReduce(MRJob):

    def mapper(self, _, line):
        prevailing_wages = []
        #f = open(r"prevailing_wage.txt", "r")
        #for line in f:
        x = line.rstrip("\n")
        #prevailing_wages.append(float(x))
        yield x,1

    def reducer(self, x, counts):
        sumwage = 0.0
        number=0.0
        average= 0.0

        for wages in counts:
            y = float(x)
            sumwage += y
            number+=1
            average = sumwage/number
            #min_wage = next(counts)
            #max_wage = min_wage
            #for cal in x:
                #min_wage = min(cal,float(min_wage))
                #max_wage = max(cal,float(max_wage))

        yield x,(number,sumwage,average)
        #yield x,(max(sumwage),min(sumwage))
        #yield x,(min_wage,max_wage)
        #yield x,(minimum_value, maximum_value)

if __name__ == '__main__':
    ProjectMapReduce.run()
```

Git hub link: https://github.com/Siddhikul27/Big-Data-Project

Video Demo link: https://drive.google.com/drive/u/1/folders/1o3JzB2tuKN_avIhvQSN92znubzNnYwdE