Summer Training Report on

Hadoop and Big Data Technologies

from

EduPristine Pvt. Ltd. New Delhi

BACHELOR OF TECHNOLOGY

COMPUTER ENGINEERING

Submitted by:

Abhinav Mudgal

11520008

Submitted to:

Ms. Minakshi Sharma

CSE Department

DEPARTMENT OF COMPUTER ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

KURUKSHETRA

Neev Knowledge Management Pvt. Ltd
702, Raaj Chambers, 115, R K. Paranjpeis Marg
Old Nagardas Road, Andheri (E
Mumbai, India – 400 069
CIN No. U80101MH2008PTC186031

EDU PRISTINE

Date: 31st July 2017

TO WHOM IT MAY CONCERN

This is to certify that **Mr. Abhinav Mudgal** S/O **Mr. Manoj Sharma**, student of Indian Institute of Information Technology, Sonepat (Mentor NIT Kurukshetra, Haryana), has completed his training from 21/05/2017 to 23/07/2017 in Big Data and Hadoop technology under the guidance of **Mr. Nitin Choudhary**. During his training period, he has been sincere and inquisitive.

Swapnil Sawant

Head of Sales department - Big Data and Hadoop

Acknowledgement

I would like to express my deepest appreciation to my trainer **Mr. Nitin Chaudhary, Technology Analyst** at ZS Associates, New Delhi for proper guidance and full support in understanding big data technologies. The understanding he had regarding the practical implications of all the desired technologies of the course was beyond words.

Furthermore, I would like to thank all my peers who came together from different companies to achieve a common goal, to learn these new technologies. Without a proper team and co-operative hands-on sessions, it would have been really difficult to complete this work.

Lastly, I would like to thank the almighty and my parents for their moral support and my friends with whom I shared my day-to-day experience and received lots of suggestions that improved my quality of work.

-Abhinav Mudgal

Declaration

I, Abhinav Mudgal, student of B.Tech. 5th Semester, studying at NIT Kurukshetra, hereby declare that the summer training report on "**Hadoop and Big data Technologies**" submitted to Computer Engineering Department, NIT Kurukshetra in partial fulfilment of Degree of Bachelor of Technology is the original work conducted by me. The information and data given in the report is authentic to the best of my knowledge. This summer training report is not being submitted to any other University for award of any other Degree, Diploma and Fellowship.

Abhinav Mudgal

NIT Kurukshetra

11520008

CO-1

Table of Contents

I. ABSTRACT

The training report aims at providing all the information regarding the technologies used for the storage, analysis and processing of the Big Data. The need, demand and importance of big data is vital to understand and has been discussed first. The challenges faced while processing such huge data items and the solutions corresponding to it are mentioned. Hadoop Distributed File System along with the processing counter-part Map Reduce are briefly explained for the understanding of the live projects that were implemented over the course of summer training. Other Advanced and desired technologies are also explained briefly. Hands-on experience on three live projects is step by step procured. Project Documentations are provided thoroughly followed by their practical implementations, procedures, commands, snap snippets and the output snippets provided as per necessity. Projects have been explained with all the requirements and possibilities before the actual implementations are shown. The effectively useful presentation of only desired utilities is displayed in the report. The report is concluded with the future work targets of the big data technologies and describing their power as fairly as possible.

II. INTRODUCTION

In pioneer days they used oxen for heavy pulling, and when one ox couldn't budge a log, they didn't try to grow a larger ox. We shouldn't be trying for bigger computers, but for more systems of computers.

--- Grace Hooper

2.1 Big Data Overview

We live in the data age. It's not easy to measure the total volume of data stored electronically, but an IDC estimate put the size of the "digital universe" at 4.4 zettabytes in 2013 and is forecasting a tenfold growth by 2020 to 44 zettabytes.1 A zettabyte is 1021 bytes, or equivalently one thousand exabytes, one million petabytes, or one billion terabytes. That's more than one disk drive for every person in the world.

This flood of data is coming from many sources

Consider the following:

- The New York Stock Exchange generates about 4−5 terabytes of data per day.
- Facebook hosts more than 240 billion photos, growing at 7 petabytes per month.
- Ancestry.com, the genealogy site, stores around 10 petabytes of data.
- The Internet Archive stores around 18.5 petabytes of data.

So, there is a lot of data out present in the digital universe. Also, the volume of data being made available is increasing exponentially. These huge chunks of data are termed as "Big data".

Where is this Data

Big data refers to large datasets which are hard to computationally analyze and process. This data is generated from multiple sources. The data that goes in the logs of google, Facebook, LinkedIn, yahoo servers is of billion users of all around the world.What are users accessing, how long the user remains in site. All the meta data sites visited, friend's list, status. Torrent downloads In Every 5 minutes granularity google gets Petabytes information in its server logs. Same goes for

Facebook, yahoo, AT&T, Airtel.The entire internet is flooded with huge amount of data and possibility of processing this data gives chance to the organizations to gather information and form better systems.

Why this data is needed

The primary concern behind every system we made is to collect and retrieve information, data. The algorithms are made to process the data and use it for the betterment of the society. If we are able to analyze more data, we can provide better solutions to the society. A great example of that can be **Internet of Things (IoT),** through which systems are communicating to each other, gathering information and creating and using big amount of data.

Besides that, Companies need such huge data for the advertisements, marketing or statistical computational purposes.

Network Companies like Airtel is India or AT&T in US monitor routers logs where information of subscriber phone id, call id is recorded to find top subscribers and install routers where traffic is more etc. Google, Amazon, eBay they get logs so that ads and products can be recommended to customers.

Challenges

The problem is not getting this big data. Problem is how to **store, process and analyze** this data.

Traditional RDBMS fails here.

The most common and abundant form in which the data is present is in the form of logs. Logs are really very important and need not to be properly stored and processed as per the given requirements of the firm or the organization. This led to switch from normal to distributed systems. RDBMS comes with a drawback of structured data processing only which can be overcame using these distributed systems and file clusters. Traditional RDBMS cannot handle the affects of these log files.Companies need such huge data for the advertisements, marketing or statistical computational purposes A typical log file may appear like follow:

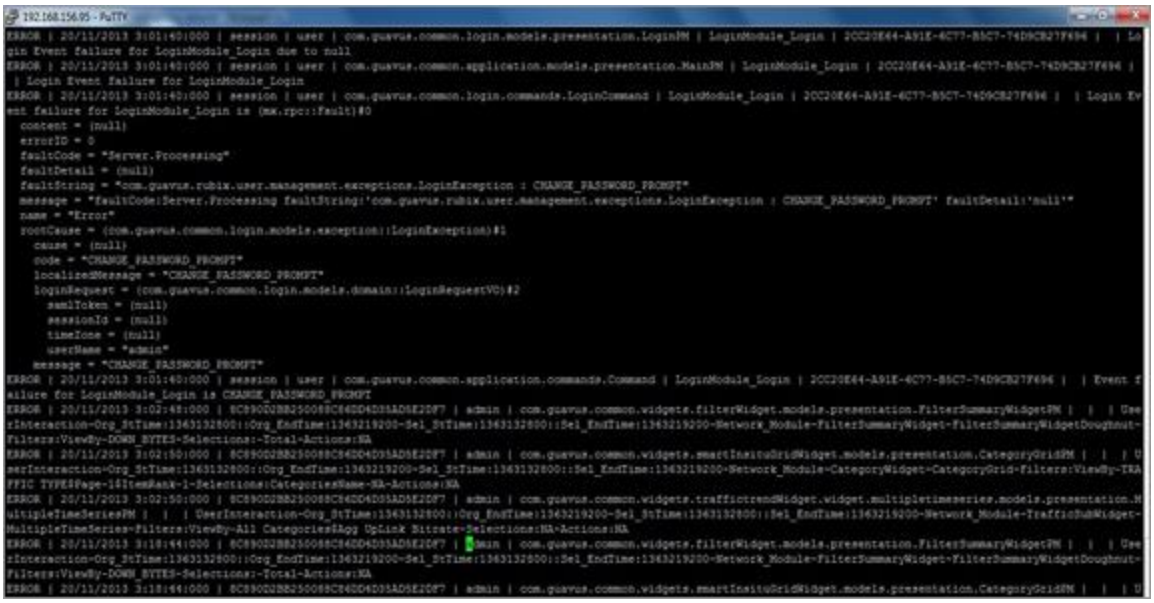

Figure 1.0

Logs are **Semi-structured** data. They do not have any default structure as in the case of relational databases. So, Traditional RDMS fails to deal with such data.

1. Data storage Problem

The data collection is so huge that it is not possible to store this concerned data on a single machine. The data is mostly of the range of petabytes. Typical Hard drives are approximately of 1 Tera byte. Even if we add external drives, we cannot store data in petabytes.

2. Processing and Analysis of Big data

To process a file or even to read a file, we need to put that file from storage device onto the Random-Access Memory. Due to inefficient primary memory, processing of huge files is not possible on a single machine.

The problem is simple: although the storage capacities of hard drives have increased massively over the years, access speeds—the rate at which data can be read from drives— have not kept up. One typical drive from 1990 could store 1,370 MB of data and had a transfer speed of 4.4 MB/s,4 so you could read all the data from a full drive in around five minutes. Over 20 years later, 1-

terabyte drives are the norm, but the transfer speed is around 100 MB/s, so it takes more than two and a half hours to read all the data off the disk.

Solution

This is a long time to read all data on a single drive—and writing is even slower. The obvious way to reduce the time is to read from multiple disks at once. Imagine if we had 100 drives, each holding one hundredth of the data. Working in parallel, we could read the data in under two minutes.

This distributed fashion of data processing can be achieved after setting up a cluster of nodes and accessing that using Hadoop distributed file system and processing through traditional MapReduce.

2.2 Hadoop Distributed File System

When a dataset outgrows the storage capacity of a single physical machine, it becomes necessary to partition it across a number of separate machines. Filesystems that manage the storage across a network of machines are called distributed filesystems. Hadoop is one such distributed file system called HDFS.

The Design of HDFS

- HDFS is a filesystem designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware.
- HDFS is built around the idea that the most efficient data processing pattern is a write-once, read-many-times pattern. A dataset is typically generated or copied from source, and then various analyses are performed on that dataset over time.
- Hadoop doesn't require expensive, highly reliable hardware. It's designed to run on clusters of commodity hardware.
- Like in a filesystem for a single disk, files in HDFS are broken into block-sized chunks (128 MB by default), which are stored as independent units. Unlike a filesystem for a single disk, a file in HDFS that is smaller than a single block does not occupy a full block's worth

of underlying storage. (For example, a 1 MB file stored with a block size of 128 MB uses 1 MB of disk space, not 128 MB.)

- An HDFS cluster has two types of nodes operating in a master−worker pattern: a name node (the master) and a number of data nodes (workers). The name node manages the filesystem namespace. It maintains the filesystem tree and the metadata for all the files and directories in the tree.

2.3 MapReduce

MapReduce provides a programming model that abstracts the problem from disk and reads and writes, transforming it into a computation over sets of keys and values for data processing.A typical example that we performed to understand MapReduce and its parallel processing was to process 1 PB data to find the most popular movie actor on twitter.We projected a word count program using map reduce to tackle such kind of problem and retrieving the results back to the disk.

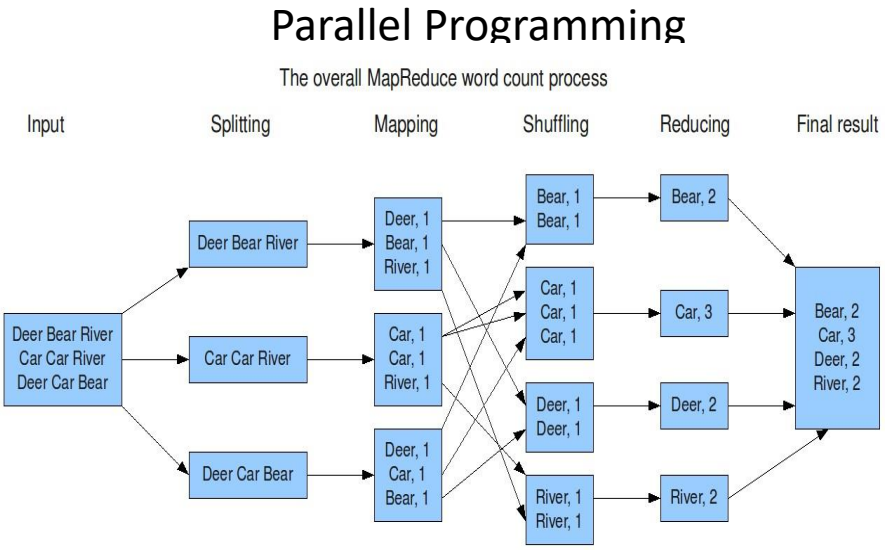

Figure 1.1

The dataset is divided into blocks and distributed across the nodes of the cluster where mapper is triggered parallelly onto concerned nodes and process them. Internally, shuffling of data based on the key value pairs is done and sent to the reducer phase. The reducer class merges or reduces the data as per the requirements and writes the results back to the disk.

2.4 Advanced Technologies

2.4.1 Pig

Apache Pig raises the level of abstraction for processing large datasets. MapReduce allows us, as the programmer, to specify a map function followed by a reduce function, but working out how to fit your data processing into this pattern, which often requires multiple MapReduce stages, can be a challenge. With Pig, the data structures are much richer, typically being multivalued and nested, and the transformations we can apply to the data are much more powerful. They include joins, for example, which are not for the faint of heart in MapReduce.

Pig turns the transformations into a series of MapReduce jobs, but as a programmer we are mostly unaware of this, which allows us to focus on the data rather than the nature of the execution

Pig is a scripting language for exploring large datasets. One criticism of MapReduce is that the development cycle is very long. Writing the mappers and reducers, compiling and packaging the code, submitting the job(s), and retrieving the results is a time-consuming business, and even with Streaming, which removes the compile and package step, the experience is still involved. Pig's sweet spot is its ability to process terabytes of data in response to a half-dozen lines of Pig Latin issued from the console.

2.4.2 Hive

One of the biggest ingredients in the Information Platform built by Jeff's team **at Facebook** was Apache Hive, **a framework for data warehousing on top of Hadoop**. Hive grew from a need to manage and learn from the huge volumes of data that Facebook was producing every day from its burgeoning social network. After trying a few different systems, the team chose Hadoop for storage and processing, since it was cost effective and met the scalability requirements

Hive is built at the top of Map Reduce and provides below features: Tools to enable easy access to data via SQL syntax, thus enabling data warehousing tasks such as extract/transform/load (ETL), reporting, and data analysis.

Query execution via various executions engines like Tez, Spark and Map Reduce. There is no specific Hive format that data must be stored, but various connectors to read and write Text, CSV data. File formats like:

Avro File Format, ORC File Format, Parquet, LZO Compression.

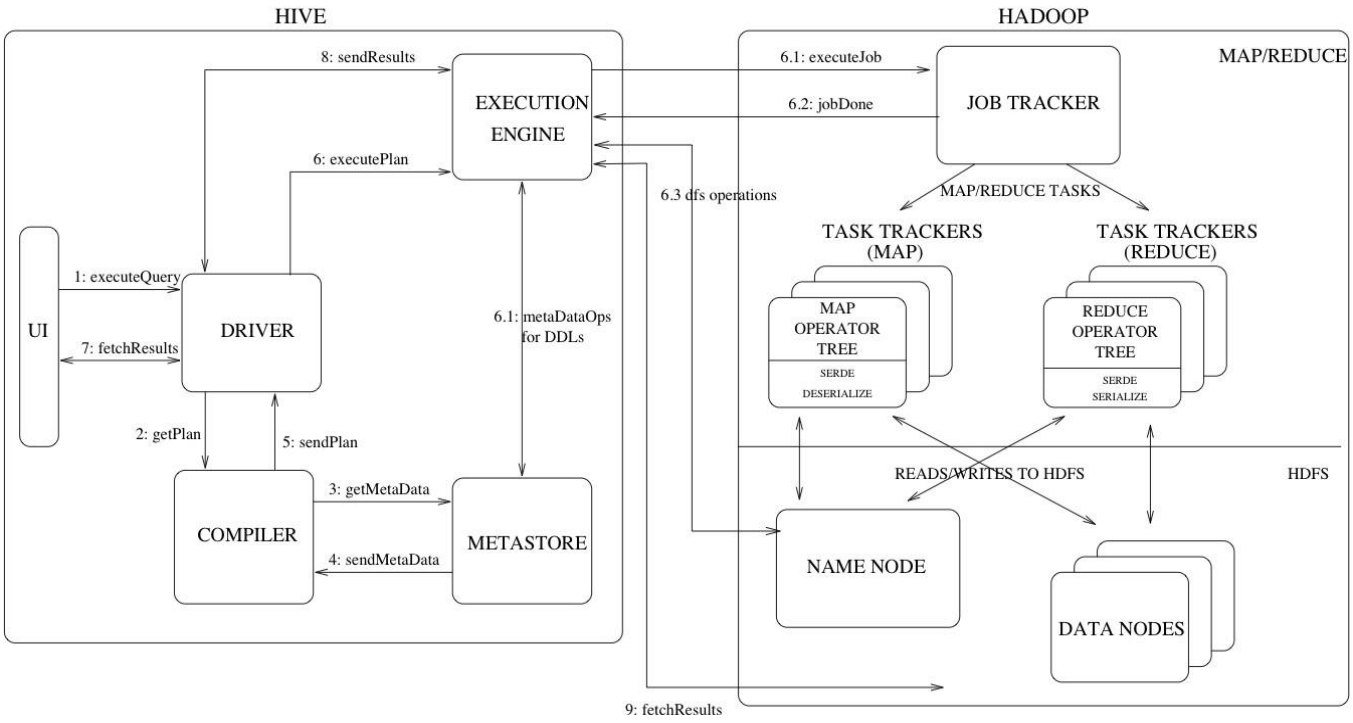

Figure 1.2

2.4.3 Spark

Apache Spark is a cluster computing framework for large-scale data processing. Unlike most of the other processing frameworks discussed before namely hive or pig, Spark does not use MapReduce as an execution engine; instead, it uses its own distributed runtime for executing work on a cluster.

Apache Spark vs MapReduce

Map Reduce is disk IO intensive. As seen in the figure 1.1, multiple stages are performed in Map Reduce processing paradigm.

Now if iterative jobs are to be performed. There are multiple map reduce stages, and each Stage involves disk I/O. In the below diagram, there will be 6 local disk I/O's and 6 HDFS I/O Operations. This makes Map Reduce SLOW.

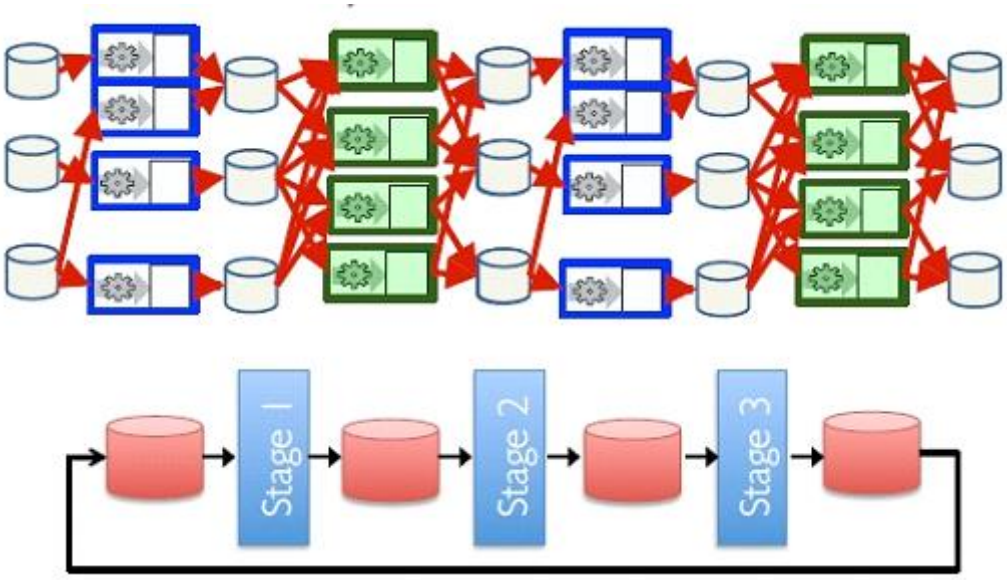

Figure 1.3

In-Memory Sharing

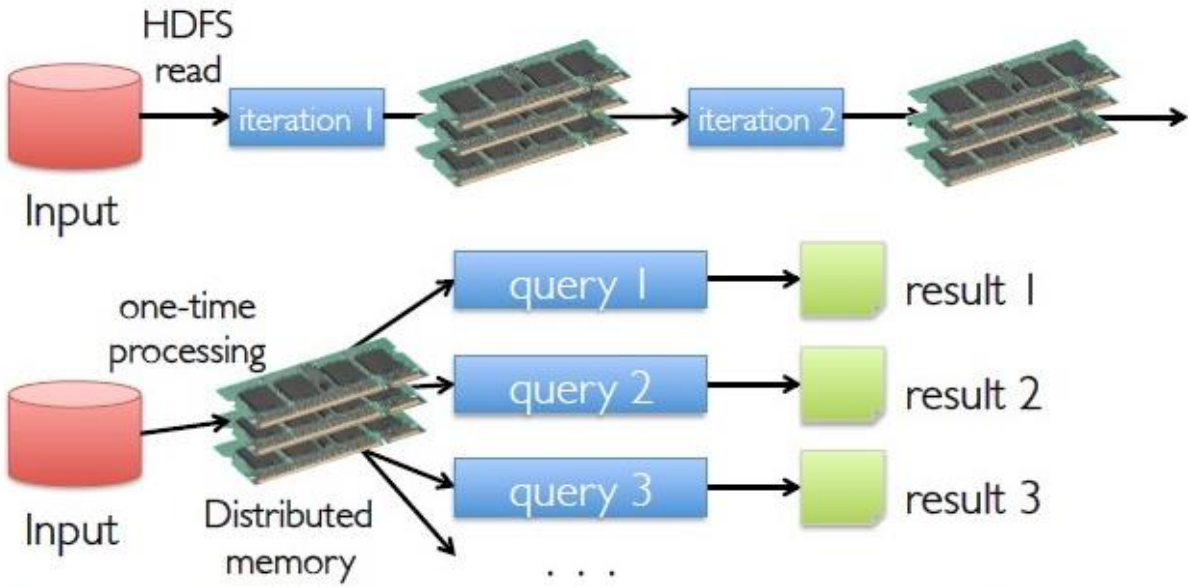

Figure 1.4

Spark on the other hand uses In-Memory sharing of source data. For faster processing and quick data sharing, memory is used. The data is cached on distributed memory of all the nodes, performed all the required queries at much faster pace.

Apache Spark Motivation

- Spark is easy to use; can write code comfortably ins; Java, Scala, Python and R.
- Spark runs programs up to 100X faster than Hadoop.
- In-Memory Caching can make a big difference. Iterative machine learning algorithms, will require multiple stages of jobs to predict the outcome.

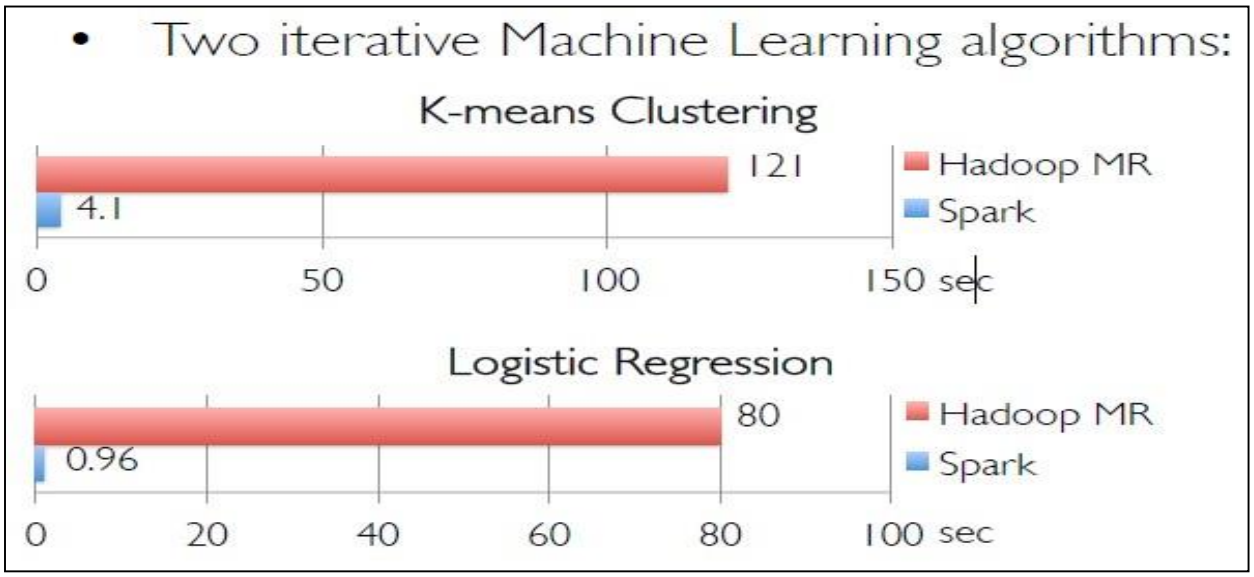

Figure 1.5

As stated from the above diagram statistically, Spark compared to Hadoop MR, runs near 100 times faster.

2.5 Other Technologies Used

2.5.1 Sqoop

Often, valuable data in an organization is stored in structured data stores such as relational database management systems (RDBMSs).Apache Sqoop is an open source tool that **allows**

users to extract data from a structured data (RDBMS) store into Hadoop for further processing.

This processing can be done with MapReduce programs or other higher-level tools such as Hive. (It's even possible to use Sqoop to move data from a database into HBase.) When the final results of an analytic pipeline are available, Sqoop can export these results back to the data store for consumption by other clients.

Sqoop Import command is used to import database from structured traditional databases onto the Hadoop.While Sqoop Export command transfers extracted distributed data back into traditional databases.

2.5.2 HBase

HBase is a distributed column-oriented database built on top of HDFS. HBase is the Hadoop application to use when you require real-time read/write random access to very large datasets. HBase is a NoSQL data store, NoSQL = "**Not Only SQL.**

Problems with Traditional RDBMS

- Traditional RDBMS is not very scalable when data reaches terabytes and petabytes realm.
- Traditional databases are designed for holding structures data. They cannot hold data like:
    - Semi structured data
    - Images
    - Network graphs
    - Documents
    - Human-Genome data
    - Health care multi media
    - GPS tracker data
    - HTML pages
    - Facebook likes, groups
    - Customer-Genome in banking, retail etc.

- NASA astronomy logs

- A table of 10000 columns in RDBMS would be unmanageable. A shot on maintenance can be cost ineffective.

Rise of NoSQL Databases

Next Generation Databases mostly addressing Business problems, being **non-relational, distributed, open-source** and **horizontally scalable.**

Main Characteristics:

Schema free, Easy replication support Simple API, Eventually Consistent (Referred as BASE – Basically Available Eventually Consistent properties) , Huge Amount data.

Consider this data: Stored in Row Stored in Column

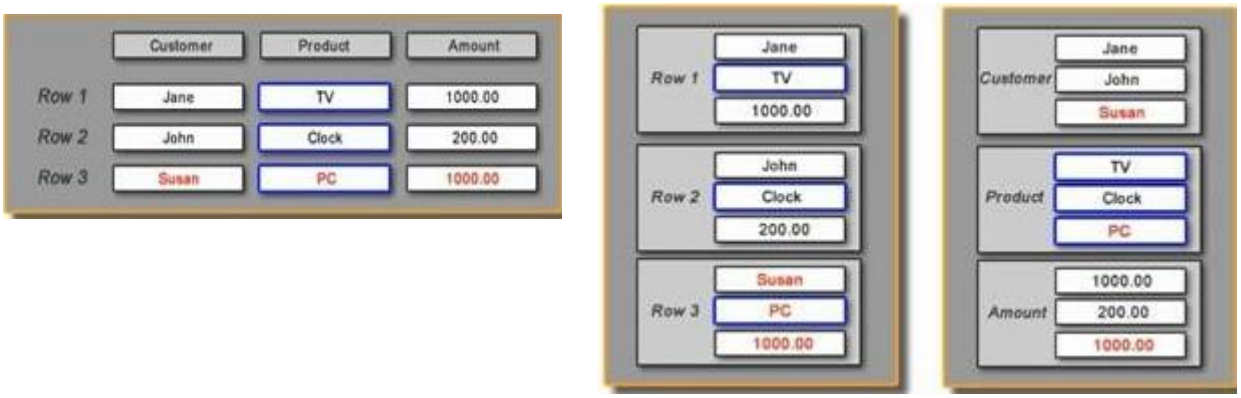

Figure 1.6

Now to get product column, in row oriented databases, all 3 columns will be scanned in each row to retrieve Product column.

In column oriented database, each column is stored separately, so Product column can be directly accessed, which is **100 times faster** than general RDBMS.

A HBase logical view of normal RDBMS database can be simply shown as follow:

Given the RDBMS:

ID	Last Name	First Name	Password	Timestamp
1234	Mudgal	Abhinav	Hello, world!	20130710
5678	Sharma	Prateek	Coding	20120825
5678	Sharma	Prateek	training	20130916

Figure 1.7

Logical View in HBase:

RowKey	**Value (CF, Qualifier, Version)**
1234	info {'lastName': 'Mudgal', 'firstName': 'Abhinav'} pwd {'password': 'Hello, world!'}
5678	info {'lastName': 'Sharma': 'firstName':'Prateek'} pwd {'password: 'training'@ts 20130916, 'password': 'Coding'@ts 20120825}

Figure 1.8

III. LIVE PROJECTS OVERVIEW

Handling of Big data is a big deal. Traditional resources that were readily available to handle datasets are not at all reliable and useful when it comes to large datasets processing and analysis. But these datasets need to be analyzed for the proper and complete utilization of some great technological achievements like Machine learning algorithms or Internet of things.

We can gather the data as much as possible and apply machine learning algorithms on it to achieve exceptional results. Or we can track huge amount of data being reciprocated across the entire electronic universe.

All such applications require the analysis and processing of data (or in major case Big Data). I have done hands-on on major technologies dealing with this data extraction and processing with optimal possible manner. The best way to learn any technology is to actually use and implement it and understand how it works.

For that, I have worked along with my team of working professionals from various companies on three major live hands-in projects with an attempt to understand Big data and its technologies.

The following are some of the major live hands-on performed during the training:

1. We performed an **Analysis of all the transactions of a Retail Company** together, which was having more than **1 million transactions record** stored in a csv file. We used big data technologies **pig and hive** for the purpose of querying and performing operations for the same on the data distributed across a 9-node cluster.

2. We also processed a **Logs file containing almost 10.5 lakh logs** using **Spark over YARN**. The python library, **pySpark** was used for the analysis. The log file was sent and distributed across a 9-node cluster and spark job was submitted over Hadoop to process.

3. We used **HBase** to implement **Customer 360 degree.** We dumped different transactions databases together from RDBMS to Hadoop using Sqoop data ingestion, later onto HBase as in a single database with a distinct column family for each database, providing customer 360-degree effect. ETL can be performed on the HBase database using pig or hive.

IV. PROJECTS DOCUMENTATION

4.1 Retail Data Analysis using Pig or Hive

Retail Transactions

Retail stores daily generate millions of transactions logs. Analyzing these logs would generate beautiful insights and improve business.

Storing these logs on traditional databases would be costly and scalability will be a big challenge.

Volume of Transactions

Stores like Walmart are spread across different locations. Daily millions of customers visit these stores and generate billions of logs.This billions of logs contribute to huge volume of data.

Velocity of Transactions

In peak hours, 1000's of transactions will happen in any given second.

1000's of transactions/sec across all stores contribute to high velocity of data.

Variety of Transactions

The most widely known varieties of data generated by transactions:

- JSON Format
- XML Format
- CSV Format

Having Huge Volume, High Velocity and variety makes this data- Big Data.

Challenges

Handling of Big data need to be tackled with challenges like of:

- Storage & Processing
- Scalability & Sharing

All of these challenges can be easily overcome using Hadoop Distributed File System.

Goals

- Processing these logs over night as a batch to understand:
- Demand of a given product.
- Trend and seasonality of sales.
- Understand performance of chain.
- Loyal Customer identification.

4.2 Log Analysis using Spark

In this digital world, humans leave digital trace at many points. In banking sector, there are ATM swipes, POS swipes, internet banking. At every point, there is information generated which are referred to as logs. In companies like amazon and flipkart, daily millions of clicks happen and each click will generate a log.

What is log?

Log is a form of data captured while an action is performed. It can be a search query log, transaction log, Nasa astronomy log, any information related to past can be considered as logs.

Logs Format

Format depends on type of logs being captured.

- User browsing logs will mostly be in text format
- Twitter logs are in JSON format
- GPS logs are in JSON format
- Machine logs will be in binary format

We on the other hand, processed logs in semi structured text format in which we first identified the structure of the log and then processed them.

```
127.0.0.1 - - [01/Aug/1995:00:00:01 -0400] "GET /images/launch-logo.gif HTTP/1.0" 200 1839
```

1: IP Address of Host Machine

2: User identity from remote machine

3: User identity from local region

4: Time stamp and time zone. Here -0400 refers to time zone.

5: get or pull request

6: end point. If we get on to flipkart and click on any link, we will be directed to that page. Anything after flipkart.com is an endpoint. E.g. https://flipkart.com/**laptops/HP**

7: HTTP protocol

8: Response code. Number starting from 2 is successful response, 3 is redirection, 4 is error caused by client, 5 is error in server.

9: bytes downloaded.

Why Log Analysis?

There is a saying "the more you understand your past, the better your future is". The same applies everywhere.

- For banking sectors, based on transaction logs, user buying patterns, customer wallet can be derived.
- In E-Commerce sectors, user browsing patterns, user similarities in browsing, recommendation, personalized pages are only possible with log analysis.
- In retail sectors, Market basket analysis, customer purchase patterns can be uncovered.

Why spark for log analysis?

Spark is chosen for following reasons:

- Interactive shell for programming
- High speed in memory computing
- Rich set of data analysis functions
- Easy of programming
- Most suited for text analytics

4.3 360degree View of Customer

Customer 360 refers to summarized information related to customer, at every digital touch point, which describes the behavior of customer, and predicts what can happen with him in future.

It can be thought on mega table, which is holding information of all products a customer holds, summary of all customers transactions, demographic features, CRM information.

Why understand your customers?

Improve the entire customer lifecycle with a customer 360-degree profile.

Any team dealing with customers must leverage technology to:

collect and analyze customer data, execute successful omni-channel campaigns, understand the customer lifecycle and influence buyers in a congested market.

Important USE CASES

Customer Acquisition & Retention

How to acquire a new customer and how to retain existing customers is a big-time challenge for any marketing team. But with customer 360, they know the DNA of customer, thereby plan accordingly. Especially in telecom sector, this is a big challenge.

Next Best offers given by companies

The Companies need to know whom to offer. Especially in retail sector, identifying loyal customers and offering them products based on previous transactions is a bigger challenge.

Having customer DNA will make it much simpler.

Customer Satisfaction

Email sent by a customer, needs an immediate response.

It's important that organizations collect every interaction in order to identify leading indicators of unhappy customers, keep their existing customers, and improve net promoter scores.

Up sell and cross sell recommendations

Identifying similar customers is on the top priority for all the industries, as it can solve upsell, cross sell, product recommendation and many other problems. To identify similar customers, one should have all possible information of customers, summarized and store digitally. Customer 360 can do that.

Hadoop for Customer 360

Hadoop can be used for implementing customer 360 database due to utility of NoSQL database i.e. HBase which can provide large datasets at one place in one go.

Also, Existing systems are costly, Not Scalable, Vendor locked and slow in processing as compared to Hadoop for implementation.

V. IMPLEMENTATION

5.1 Retail Data Analysis

5.1.1 Flow Diagram of the Process

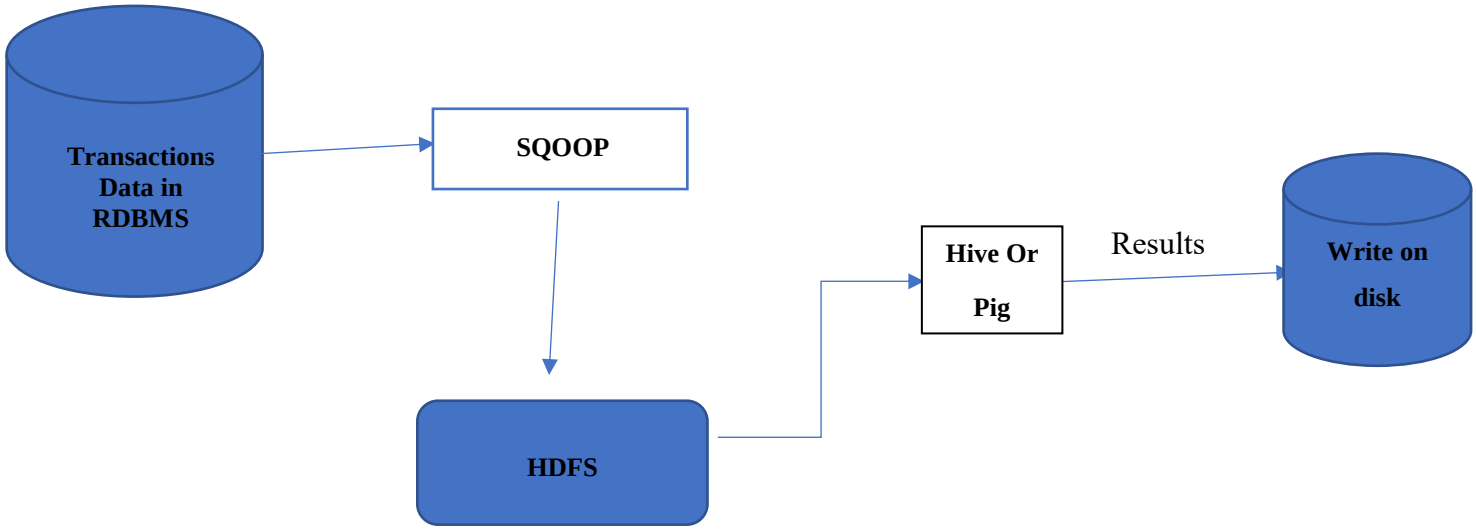

Figure 2.0

Transactions database – meta data

```
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| id               | varchar(20) | YES  |     | NULL    |       |
| chain            | varchar(20) | YES  |     | NULL    |       |
| dept             | varchar(20) | YES  |     | NULL    |       |
| category         | varchar(20) | YES  |     | NULL    |       |
| company          | varchar(20) | YES  |     | NULL    |       |
| brand            | varchar(20) | YES  |     | NULL    |       |
| date1            | varchar(10) | YES  |     | NULL    |       |
| productsize      | int(11)     | YES  |     | NULL    |       |
| productmeasure   | varchar(10) | YES  |     | NULL    |       |
| purchasequantity | int(11)     | YES  |     | NULL    |       |
| purchaseamount   | float       | YES  |     | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
11 rows in set (0.00 sec)
```

Figure 2.1

Snapshot of CSV file

	A	B	C	D	E	F	G	H	I	J	K	L
1048552	97170143	14	33	3305	1.06E+09	5554	3/30/2013	16	OZ	1	3.79	
1048553	97170143	14	9	902	1.01E+08	45039	3/30/2013	30	OZ	1	7.99	
1048554	97170143	14	26	2628	1.02E+08	15704	3/30/2013	6	RL	1	6.99	
1048555	97170143	14	29	2908	1.04E+08	15487	3/30/2013	30	OZ	1	1.89	
1048556	97170143	14	63	6315	1.02E+08	10786	3/30/2013	128	OZ	1	2.79	
1048557	97170143	14	21	2119	1.08E+08	16756	4/15/2013	96	OZ	1	5	
1048558	97170143	14	5	501	1.02E+08	3331	4/15/2013	21.6	OZ	1	5.39	
1048559	97170143	14	63	6321	1.04E+08	15567	4/15/2013	10	OZ	1	3	
1048560	97170143	14	99	9909	1.08E+08	12908	4/15/2013	16	OZ	1	5.49	
1048561	97170143	14	9	902	1.03E+08	17248	4/15/2013	18	OZ	1	4.79	
1048562	97170143	14	56	5604	1.07E+08	469	4/15/2013	8	OZ	1	6.99	
1048563	97170143	14	99	9904	1.07E+09	5278	4/15/2013	16	OZ	2	2.96	
1048564	97170143	14	5	501	1.02E+08	6283	4/15/2013	16.1	OZ	2	5.98	
1048565	97170143	14	45	4509	1.02E+08	28503	4/15/2013	32	OZ	1	13.98	
1048566	97170143	14	8	818	1.07E+08	82520	4/15/2013	25	OZ	1	4.99	
1048567	97170143	14	36	3604	1.02E+08	15704	4/15/2013	72	OZ	1	1.5	
1048568	97170143	14	21	2108	1.02E+08	15704	4/15/2013	64	OZ	1	2.49	
1048569	97170143	14	63	6305	1.05E+08	18799	4/15/2013	59	OZ	2	6	
1048570	97170143	14	9	901	1.02E+08	15704	4/15/2013	8	OZ	1	2.69	
1048571	97170143	14	99	9909	1.08E+08	12908	4/21/2013	16	OZ	1	4.99	

Figure 2.2

5.1.2 Working

1. Logged into Remote Server IP 101.53.130.144

2. Accessed MySQL of remote database through **mysql -h 101.53.130.146 -u mudgal_abhinav19 --local-infile.**

3. Meta data created as per the given comma separated valued file.

CREATE TABLE transactions(id varchar(20),chain varchar(20), dept varchar(20),category varchar(20), company varchar(20), brand varchar(20), date1 varchar(10), productsize int, productmeasure varchar(10), purchasequantity int, purchaseamount FLOAT);

4. Loaded the data in sql from transactions.csv

LOAD DATA LOCAL INFILE '/home/mudgal_abhinav19/transactions_practice.csv' INTO TABLE transactions FIELDS TERMINATED BY ',' ENCLOSED BY'"' LINES TERMINATED BY '\n';

1. Sqoop used to import data from mysql of the remote server to the 9-node cluster of Edu Pristine containing Hadoop distributed File System.

sqoop import --connect jdbc:mysql://101.53.130.146/mudgal_abhinav19 --username mudgal_abhinav19 -P --table transactions -m 1 --target-dir TransactionsData

Directory gets created on hdfs TransactionsData with data loaded in it.

-rw-r--r-- 3 mudgal_abhinav19 labusers 0 2017-07-23 14:49 TransactionsData/_SUCCESS

-rw-r--r-- 3 mudgal_abhinav19 labusers 60317773 2017-07-23 14:49 TransactionsData/part-m-00000

Now if remove -m 1 and split on id , 4 mappers gets called

sqoop import --connect jdbc:mysql://101.53.130.146/mudgal_abhinav19 --username mudgal_abhinav19 -P --table transactions -split-by id --target-dir TransactionsData

-rw-r--r-- 3 mudgal_abhinav19 labusers 0 2017-07-23 15:17 TransactionsData/_SUCCESS

-rw-r--r-- 3 mudgal_abhinav19 labusers 7880230 2017-07-23 15:16 TransactionsData/part-m-00000

-rw-r--r-- 3 mudgal_abhinav19 labusers 6175607 2017-07-23 15:17 TransactionsData/part-m-00001

-rw-r--r-- 3 mudgal_abhinav19 labusers 20992989 2017-07-23 15:16 TransactionsData/part-m-00002

-rw-r--r-- 3 mudgal_abhinav19 labusers 25268947 2017-07-23 15:16 TransactionsData/part-m-00003

2. Now can import directly to hive using sqoop.

sqoop import --connect jdbc:mysql://101.53.130.146/mudgal_abhinav19 --username mudgal_abhinav19 -password DszEyFSZhaRQ92O --table transactions --hive-import --hive-database abhi --hive-table transactions_staging -m 1

3. After this setup can fire any query on the data to process it accordingly.

5.1.3 Screenshots with outputs:

```
-- Top 10 customers

select id, sum(purchaseamount) as custSpendings from transactions_production
group by id
sort by custSpendings DESC
limit 10;
```

```
86252    53592.89999999957
86246    52828.11999999952
57132131         20863.29000000036
83868868         15302.169999999785
67957375         14684.609999999933
58237989         14611.269999999817
91998805         14572.45999999987
94244413         14302.60999999986
54590006         13740.629999999846
56544981         13101.059999999881
Time taken: 178.254 seconds
```

Figure 3.0

```
-- chain wise sales
select chain, sum(purchaseamount), sum(purchasequantity) from transactions_production
group by chain;
```

```
OK
14      365785.2999998729       98427
15      1077677.4699990302      319263
17      394737.39999981265      123889
18      693190.209999508        210325
2       3814.6399999999644      1194
20      291686.0799999232       92454
205     106421.01999999923      34937
3       208104.02999998012      65062
4       767159.3599993604       245482
58      10223.759999999922      4438
88      324290.9899998967       104758
95      441778.4099997955       145003
Time taken: 62.781 seconds
```

Figure 3.1

```
-- top 10 brands
select brand, sum(purchaseamount) as custSpendings from transactions_production
group by brand
sort by custSpendings DESC
limit 10;
```

```
OK
15704   449154.8199997314
10786   221273.36999998149
0       102354.54000000849
12908   61599.71999999789
16397   52987.13999999947
30626   44524.7600000001
13310   37686.130000001845
33170   36812.940000002265
17286   36278.900000000904
9886    30633.260000002185
Time taken: 141.856 seconds
```

Figure 3.2

```
-- top 10 companies
select company, sum(purchaseamount) as custSpendings from transactions_production
group by company
sort by custSpendings DESC
limit 10;
```

```
Total MapReduce CPU Time Spent: 19 seconds 760 msec
OK
102113020       761962.2299992407
107989373       136652.57000001118
103700030       115869.43000001504
107675070       109698.45000000006
102840020       76357.6199999992
104900040       58961.5100000003
101600010       58111.500000000546
104400040       51936.17999999996
101200010       51542.92999999974
103800030       47719.360000000524
Time taken: 134.221 seconds
```

Figure 3.3

```
-- Chain Year Monthly Sales
select chain, split(date1,'/')[2] as year1, split(date1,'/')[0] as month1,
sum(purchaseamount) as totalsales from transactions_production
group by chain,split(date1,'/')[0],split(date1,'/')[2];
```

```
OK
14      2013    1       25989.100000000908
14      2012    10      24914.220000000976
14      2012    11      26941.65000000088
14      2012    12      34061.4400000014
14      2013    2       24553.660000000826
14      2012    3       22811.400000000634
14      2013    3       25149.64000000094
14      2012    4       23113.890000000614
14      2013    4       17588.63999999994
14      2012    5       26469.71000000088
14      2013    5       10979.339999999844
14      2012    6       24692.90000000072
14      2013    6       8059.429999999861
14      2012    7       21334.34000000049
14      2013    7       543.8000000000003
14      2012    8       23286.67000000069
14      2012    9       25295.47000000095
15      2013    1       77346.02000000096
15      2012    10      77011.47000000112
15      2012    11      76465.90000000053
15      2012    12      97227.45000000458
15      2013    2       70020.98999999887
15      2012    3       69659.74999999933
15      2013    3       77026.080000001
```

Figure 3.4

5.2 Log Analysis Using Spark

5.2.1 General Stages of Data Analysis

- Data Understanding
- Data Cleansing and preprocessing
- Data analysis
- Sharing result

Snap snippet of logfile

```
1043145  www-c4.proxy.aol.com - - [22/Aug/1995:23:59:26 -0400] "GET /images/ksclogo-medium.gif HTTP/1.0" 304 0
1043146  www-c4.proxy.aol.com - - [22/Aug/1995:23:59:27 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 304 0
1043147  asn18.whidbey.net - - [22/Aug/1995:23:59:28 -0400] "GET /icons/blank.xbm HTTP/1.0" 304 0
1043148  asn18.whidbey.net - - [22/Aug/1995:23:59:28 -0400] "GET /icons/menu.xbm HTTP/1.0" 304 0
1043149  www-c4.proxy.aol.com - - [22/Aug/1995:23:59:28 -0400] "GET /images/MOSAIC-logosmall.gif HTTP/1.0" 200 363
1043150  asn18.whidbey.net - - [22/Aug/1995:23:59:28 -0400] "GET /icons/image.xbm HTTP/1.0" 304 0
1043151  tty52.tcl.nd.edu - - [22/Aug/1995:23:59:29 -0400] "GET /history/apollo/apollo-8/apollo-8.html HTTP/1.0" 200 3625
1043152  199.4.102.54 - - [22/Aug/1995:23:59:30 -0400] "GET /ksc.html HTTP/1.0" 200 7087
1043153  www-c4.proxy.aol.com - - [22/Aug/1995:23:59:30 -0400] "GET /images/USA-logosmall.gif HTTP/1.0" 200 234
1043154  tty52.tcl.nd.edu - - [22/Aug/1995:23:59:30 -0400] "GET /history/apollo/apollo-8/apollo-8-patch-small.gif HTTP/1.0" 200 11326
1043155  inet-tis.toshiba.co.jp - - [22/Aug/1995:23:59:32 -0400] "GET /history/apollo/images/footprint-small.gif HTTP/1.0" 200 18149
1043156  inet-tis.toshiba.co.jp - - [22/Aug/1995:23:59:32 -0400] "GET /images/KSC-logosmall.gif HTTP/1.0" 200 1204
1043157  asn18.whidbey.net - - [22/Aug/1995:23:59:36 -0400] "GET /icons/unknown.xbm HTTP/1.0" 304 0
1043158  internet-gw.watson.ibm.com - - [22/Aug/1995:23:59:37 -0400] "GET /history/apollo/a-001/a-001.html HTTP/1.0" 200 1236
1043159  199.4.102.54 - - [22/Aug/1995:23:59:38 -0400] "GET /images/ksclogo-medium.gif HTTP/1.0" 200 5866
1043160  pm5-29.tvs.net - - [22/Aug/1995:23:59:38 -0400] "GET /shuttle/countdown/liftoff.html HTTP/1.0" 200 5222
```

Figure 4.0

5.2.2 Project Working

1. Data understanding: Log format – Identified the type of log file and its specifications. Refer to log analysis documentation section.

2. Data cleansing and preprocessing – Extracted the structure out of the given semi structured data in the log format. One way to split on space was not possible as [] in timestamp and “” still remain unprocessed.

So, Used Regular expressions of Python to extract the required pattern.

APACHE_ACCESS_LOG_PATTERN = '^(\S+) (\S+) (\S+) \[([\w:/]+\s[+\-]\d{4})\] "(\S+) (\S+)\s*(\S*)\s*" (\d{3}) (\S+)'

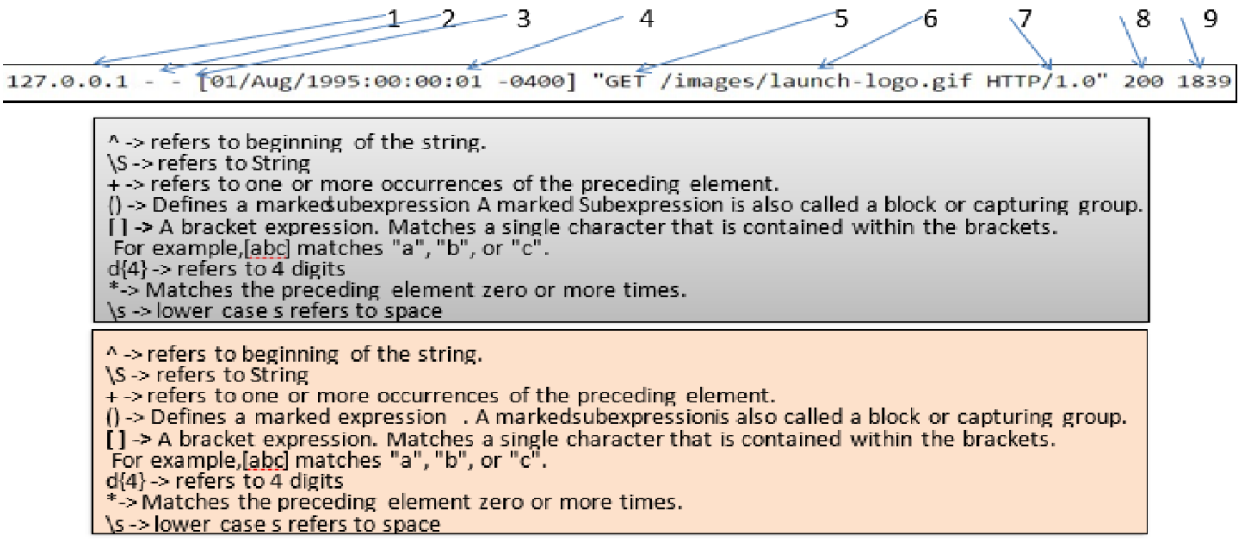

Figure 4.1

Lab Processing

Imported the entire log file onto the HDFS using Hadoop commands on PUTTY:

hadoop fs -mkdir sparkProject

hadoop fs -put ~/sparkProject/apache.access.log.PROJECT sparkProject

hadoop fs -ls sparkProject

Using WinSCP, transferred all the python scripts on the cluster.Submitted Initial Loader job to parse the log file structure using regular expressions.**spark-submit 1_IntialLoader.py > 1_linecount**Ran all the desired operations (Python programs) onto the log file through a Runner program as follows. As the only thing we have to work is the project code along with the spark jobs, Hadoop namenode processing and commands. Python scripts are run over the cluster of 9 nodes and in memory caching is achieved using spark.The output is stores back to the local edge nodes after the entire processing Extracted the structure out of the given semi structured data in the log format. One way to split on space was not possible as [] in timestamp and “” still remain unprocessed.

5.3 Customer 360 DNA

5.3.1 Flow diagram for the Overall Process

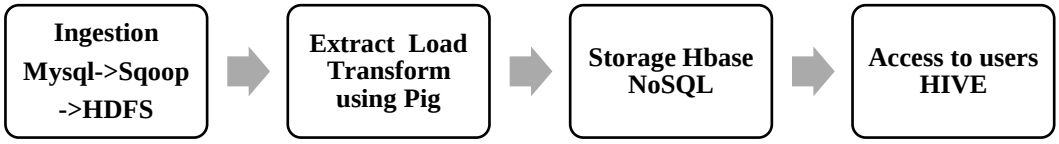

Figure 5.0

5.3.2 Project Work

1. We have been provided with six datasets in excel sheets AVRO format:

Credit card: Credit card account details.

Credit card trx: Credit card transactions details.

Demographics: Customer demographic details.

Deposit: Deposit account details.

Loan: Loan account details.

Savings: Savings account details

2. Loaded the data into MySQL database first.

One out of all such queries is like

CREATE TABLE savingsaccount(customerid varchar(11), savingsid varchar(11),avgbalance int(10));

show tables;

desc savingsaccount;

LOAD DATA local INFILE "/home/mudgal_abhinav19/1.DataSets/savingsaccount.csv"

INTO TABLE savingsaccount
COLUMNS TERMINATED BY ','
LINES TERMINATED BY '\n' IGNORE 1 LINES;

3. Data Ingestion. Sqoop used to ingest data from MySQL to HDFS as in previous projects.
4. Created a Customer 360 Mega Table HBase with the following column families:

- Demographics
- Savings
- Loan
- Credit
- Deposit
- Credittrxsumary

create 'customer360_mudgal_abhinav19', 'demographics', 'savings', 'loan', 'credit', 'deposit', 'credittrxsummary'
list 'customer.*'
describe 'customer360_mudgal_abhinav19'

5. Extracted all the avro files using pig and dumped into HBase.
One such extraction was as follows:

#----- Pig Script----------

demographics = LOAD 'cust360/demographics/' USING AvroStorage();
demographics = FOREACH demographics GENERATE customerid,registrationdate, age, gender, occupation, income;
describe demographics;
illustrate demographics;
STORE demographics INTO 'hbase://customer360_mudgal_abhinav19' USING org.apache.pig.backend.hadoop.hbase.HBaseStorage(
'demographics:registrationdate

```
 demographics:age
 demographics:gender
 demographics:occupation
 demographics:income'
);
```

After extraction, the 6 datasets were altogether present in the same column family of the HBase. This ensured the Customer 360 implementation of all the workings of a particular customer which can be tracked using transformations performed through hive onto the HBase.

VI. CONCLUSION

In April 2008, Hadoop broke a world record to become the fastest system to sort an entire terabyte of data. Running on a 910-node cluster, Hadoop sorted 1 terabyte in 209 seconds. In November of the same year, Google reported that its MapReduce implementation sorted 1 terabyte in 68 seconds.15 Then, in April 2009, it was announced that a team at Yahoo! had used Hadoop to sort 1 terabyte in 62 seconds.

The trend led to current stage of sorting 100 terabytes of data in only 1,406 seconds using Apache Spark.

Through these facts, it is clear how far we have excelled and such great power these data utilities incorporate into the world.

Many big projects like Biological Data Science: **Saving lives with Software**, or composable data at cerner by integrating Healthcare Data were made possible only through Hadoop.

In a nutshell, Hadoop provides a reliable, scalable and affordable platform for us to store and process big data.

Not only Hadoop, but all these technologies are recently emerging and exponentially growing with the days. Every big enterprise is switching its technologies from old models like mainframes to these splendid technologies. Not only because of their modern and easier workflows but more because their ability to expand in the proper and larger manner. The growth with which these technologies are spreading is staggering

The power of this data is so gigantic that if the right data is extracted properly and conveniently used for the analysis through traditional ways or using machine learning algorithms, it can lead the humanity to some place which is beyond our current imaginations.

VII. REFERENCES

[1] Hadoop-The.Definitive.Guide_4.edition_a_Tom.White_April-2015

[2] Python for DataScience by Jose Portilla -udemy/bootcamp

[3] The World's Technological Capacity to Store, Communicate, and Compute- Martin Hilbert

[4] *"Survey on Big Data Using Data Mining"* (PDF). International Journal of Engineering Development and Research. 2015

[5] EduPristine Working Journal and lab manual.