DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

B.E / SEM VII / REV 2019 'C SCHEME' / CSE-(AI&ML)
Academic Year: 2023-24

NAME	**PRATHAMESH S. CHIKANKAR**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML11**
SUBJECT	**BIG DATA ANALYTICS LAB**
COURSE CODE	**CSL702**
PRACTICAL NO.	
DOP	
DOS	

Experiment – 1

Aim - Hadoop HDFS Practical: -HDFS Basics, Hadoop Ecosystem Tools Overview. -Installing Hadoop. -Copying File to Hadoop. -Copy from Hadoop File system and deleting file. -Moving and displaying files in HDFS. -Programming exercises on Hadoop.

Theory:

What Is Hadoop?

Hadoop is an open-source software platform designed to store and process quantities of data that are too large for just one particular device or server. Hadoop's strength lies in its ability to scale across thousands of commodity servers that don't share memory or disk space.

Hadoop delegates tasks across these servers (called "worker nodes" or "slave nodes"), essentially harnessing the power of each device and running them together simultaneously. This is what allows massive amounts of data to be analyzed: splitting the tasks across different locations in this manner allows bigger jobs to be completed faster.

Hadoop can be thought of as an ecosystem—it's comprised of many different components that all work together to create a single platform. There are two key functional components within this ecosystem: The storage of data (Hadoop Distributed File System, or HDFS) and the framework for running parallel computations on this data (MapReduce). Let's take a closer look at each.

Hadoop Distributed File System (HDFS)

HDFS is the "secret sauce" that enables Hadoop to store huge files. It's a scalable file system that distributes and stores data across all machines in a Hadoop cluster (a group of servers). Each HDFS cluster contains the following:

- NameNode: Runs on a "master node" that tracks and directs the storage of the cluster.
- DataNode: Runs on "slave nodes," which make up the majority of the machines within a cluster. The NameNode instructs data files to be split into blocks, each of which are replicated three times and stored on machines across the cluster. These replicas ensure the entire system won't go down if one server fails or is taken offline—known as "fault tolerance."
- Client machine: neither a NameNode or a DataNode, Client machines have Hadoop installed on them. They're responsible for loading data into the cluster, submitting MapReduce jobs and viewing the results of the job once complete.

MapReduce

MapReduce is the system used to efficiently process the large amount of data Hadoop stores in HDFS. Originally created by Google, its strength lies in the ability to divide a single large data

processing job into smaller tasks. All MapReduce jobs are written in Java, but other languages can be used via the Hadoop Streaming API, which is a utility that comes with Hadoop.

- JobTracker: The JobTracker oversees how MapReduce jobs are split up into tasks and divided among nodes within the cluster.
- TaskTracker: The TaskTracker accepts tasks from the JobTracker, performs the work and alerts the JobTracker once it's done. TaskTrackers and DataNodes are located on the same nodes to improve performance.

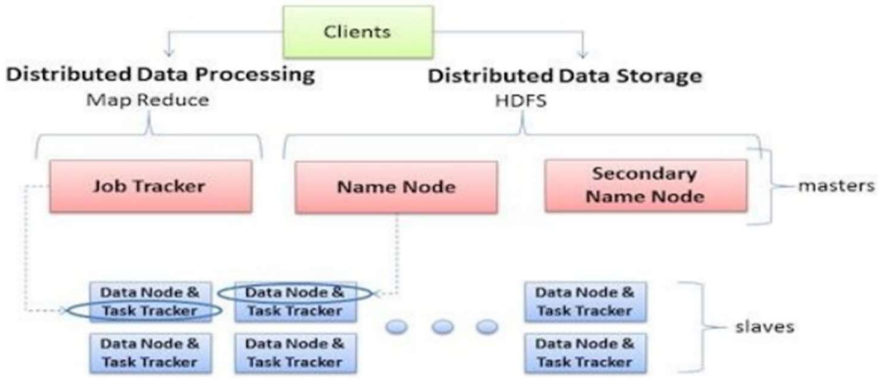
A good way to understand how MapReduce works is with playing cards.

Each deck of cards contains the following:

- Four suits: diamonds, clubs, hearts and spades.
- Numeric cards: 2, 3, 4, 5, 6, 7, 8, 9, 10.
- Non-numeric cards: ace, king, queen, jack and the jokers.

To understand how MapReduce works, imagine that your goal is to add up the card numbers for a single suit. To do this, you must sort through an entire deck of cards, one by one. As you do so, you will ‑map‖ them by separating the cards into piles according to suit.

In this example, non-numeric cards will represent ‑bad data,‖ or data we want to exclude from the results of this particular MapReduce job. As you go through the deck, you ignore the ‑bad data‖ by setting it aside from the data you do want to include.



Data locality: An important concept with HDFS and MapReduce, data locality can best be described as ‑bringing the compute to the data.‖ In other words, whenever you use a MapReduce program on a particular part of HDFS data, you always want to run that program on the node, or machine, that actually stores this data in HDFS. Doing so allows processes to be run much faster, since it prevents you from having to move large amounts of data around.

Yet Another Resource Negotiator (YARN):

YARN is an updated way of handling the delegation of resources for MapReduce jobs. It takes the place of the JobTracker and TaskTracker. In our house example, if JobTracker and TaskTracker can be thought of as the foreman, YARN is a foreman with an MBA—it's a more advanced way of carrying out MapReduce jobs.

It also gives you added abilities, such as the ability to work with frameworks other than MapReduce and to translate jobs developed in languages other than Java.

HBase

HBase is a columnar database management system that is built on top of Hadoop and runs on HDFS. Like MapReduce, HBase applications are written in Java, as well as other languages via their Thrift database, which is a framework that allows cross-language services development. The key difference between MapReduce and HBase is that HBase is intended to work with random workloads.

Hive

MapReduce jobs are often written in Java. But not everyone using Hadoop knows Java —the preferred syntax is SQL, which is essentially the ‑lingua franca‖ between all programming languages in the BI/big data space.

Installation Steps –

Following are steps to Install Apache Hadoop on Ubuntu 14.04

Step1. Install Java (OpenJDK) - Since hadoop is based on java, make sure you have java jdk installed on the system. Please check the version of java (It should be 1.7 or above it)

```
$ java –version
```

If it returns "The program java can be found in the following packages", If Java isn't been installed yet, so execute the following command:

```
$sudo apt-get install default-jdk
```

Step2: Configure Apache Hadoop

1. Open bashrc in gedit mode

```
$sudo gedit ~/.bashrc
```

2. Set java environment variable

```
export JAVA_HOME=/usr/jdk1.7.0_45/
```

3. Set Hadoop environment variable

```
export HADOOP_HOME=/usr/Hadoop 2.6/
```

4. Apply environment variables

```
$source ~/.bashrc
```

Step3: Install eclipse

Step4: Copy Hadoop plug-ins such as

- **hadoop-eclipse-kepler-plugin-2.2.0.jar**
- **hadoop-eclipse-kepler-plugin-2.4.1.jar**
- **hadoop-eclipse-plugin-2.6.0.jar** from **release** folder of **hadoop2x-eclipse-plugin-master to eclipse plugins**

Step5: In **eclipse,** start new **MapReduce project**

```
File->new->other->MapReduce project
```

Step 6: Copy Hadoop packages such as **commons-io-2.4.jar commons-lang3-3.4.jar** in **src** file of MapReduce project

Step 7: Create Mapper, Reducer, and driver

```
Inside a project->src->File->new->other->Mapper/Reducer/Driver
```

Step 8: Copy Log file **log4j.properties** from **src** file of hadoop in src file of MapReduce project

Hadoop is powerful because it is extensible and it is easy to integrate with any component. Its popularity is due in part to its ability to store, analyze and access large amounts of data, quickly and cost effectively across clusters of commodity hardware. Apache Hadoop is not actually a single product but instead a collection of several components. When all these components are merged, it makes the Hadoop very user friendly.

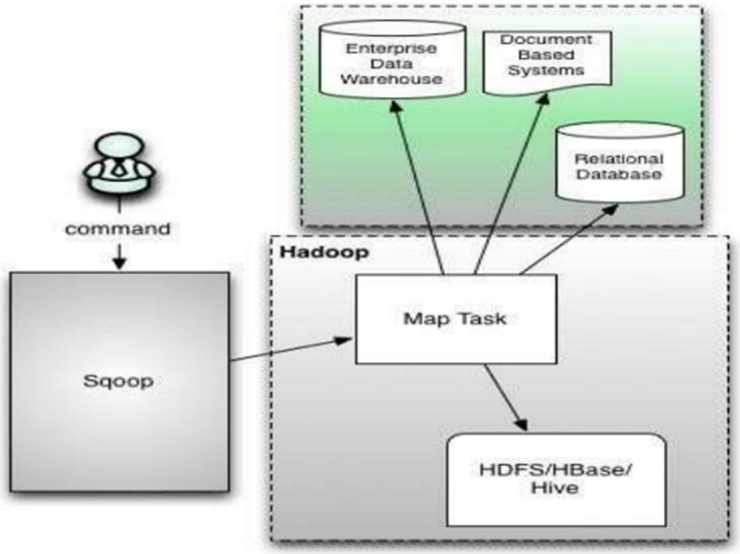
Experiment – 2

Aim - Use of Sqoop tool to transfer data between Hadoop and relational database servers. a. Sqoop - Installation. b. To execute basic commands of Hadoop eco system component Sqoop.

Theory:

What is Sqoop?
Apache Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and external datastores such as relational databases, enterprise data warehouses.
Sqoop is used to import data from external datastores into Hadoop Distributed File System or related Hadoop eco-systems like Hive and HBase. Similarly, Sqoop can also be used to extract data from Hadoop or its eco-systems and export it to external datastores such as relational databases, enterprise data warehouses. Sqoop works with relational databases such as Teradata, Netezza, Oracle, MySQL, Postgres etc.

Why is Sqoop used?
For Hadoop developers, the interesting work starts after data is loaded into HDFS. Developers play around the data in order to find the magical insights concealed in that Big Data. For this, the data residing in the relational database management systems need to be transferred to HDFS, play around the data and might need to transfer back to relational database management systems. Sqoop automates most of the process, depends on the database to describe the schema of the data to be imported. Sqoop uses MapReduce framework to import and export the data, which provides parallel mechanism as well as fault tolerance. Sqoop makes developers life easy by providing command line interface. Developers just need to provide basic information like source, destination and database authentication details in the sqoop command. Sqoop takes care of remaining part. The scoop architecture is as follows:



Implementation:

Step 1- Sqoop Installation

However, to extract the Sqoop tarball and move it to ‒/usr/lib/sqoop‖ directory we use the following command.

$tar -xvf sqoop-1.4.4.bin_hadoop-2.0.4-alpha.tar.gz

$ su

password:

mv sqoop-1.4.4.bin_hadoop-2.0.4-alpha /usr/lib/sqoop

#exit

Step 2- Configuring bashrc

Also, by appending the following lines to ~/.bashrc file we have to set up the Sqoop environment

#Sqoop

export SQOOP_HOME=/usr/lib/sqoop export PATH=$PATH:$SQOOP_HOME/bin

Now, to execute ~/.bashrc file we use the following command.

$ source ~/.bashrc

Step 3: Configuring Sqoop

While, we need to edit the sqoop-env.sh file, that is placed in the $SQOOP_HOME/conf directory, in order to configure Sqoop with Hadoop. Now, using the following command redirect to Sqoop config directory and copy the template file

$ cd $SQOOP_HOME/conf

$ mv sqoop-env-template.sh sqoop-env.sh

Also, open sqoop-env.sh and edit the following lines

export HADOOP_COMMON_HOME=/usr/local/hadoop

export HADOOP_MAPRED_HOME=/usr/local/hadoop

Step 4: Download and Configure MySQL-connector-java

From the following link, we can download MySQL-connector-java-5.1.30.tar.gz file.

In adddition, to extract MySQL-connector-java tarball and move MySQL-connector-java-5.1.30-bin.jar to /usr/lib/sqoop/lib directory we use the following command.

$ tar -zxf mysql-connector-java-5.1.30.tar.gz

$ su

password:

cd mysql-connector-java-5.1.30

mv mysql-connector-java-5.1.30-bin.jar /usr/lib/sqoop/lib

Step 5: Verifying Sqoop

At last, to verify the Sqoop version we use the following command.

$ cd $SQOOP_HOME/bin

$ sqoop-version

Expected output

14/12/17 14:52:32 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5

Sqoop 1.4.5 git commit id 5b34accaca7de251fc91161733f906af2eddbe83

Compiled by abe on Fri Aug 1 11:19:26 PDT 2014

Hence, in this way Sqoop installation is complete.

Basic Commands and Syntax for Sqoop

1. Sqoop-Import

Sqoop import command imports a table from an RDBMS to HDFS. Each record from a table is considered as a separate record in HDFS. Records can be stored as text files, or in binary representation as Avro or SequenceFiles.

Generic Syntax:

$ sqoop import (generic args) (import args)

The Hadoop specific generic arguments must precede any import arguments, and the import arguments can be of any order.

2. Importing a Table into HDFS

Syntax:

```
$ sqoop import --connect --table --username --password --target-dir
```

--connect	Takes JDBC url and connects to database
--table	Source table name to be imported
--username	Username to connect to database
--password	Password of the connecting user
--target-dir	Imports data to the specified directory

3. Importing Selected Data from Table

Syntax:

```
$ sqoop import --connect --table --username --password --columns --where
```

--columns	Selects subset of columns
--where	Retrieves the data which satisfies the condition

4. Importing Data from Query

Syntax:

```
$ sqoop import --connect --table --username --password --query
```

--query	Executes the SQL query provided and imports the results

5. Incremental Exports

Syntax:

```
$ sqoop import --connect --table --username --password --incremental --check-column --last-value
```

Sqoop import supports two types of incremental imports:

1. Append
2. Lastmodified.

6. Sqoop-Eval

Sqoop-eval command allows users to quickly run simple SQL queries against a database and the results are printed on to the console. Generic Syntax:

```
$ sqoop eval (generic args) (eval args)
```

7. Sqoop-List-Database

Used to list all the database available on RDBMS server. Generic Syntax:

```
$ sqoop list-databases (generic args) (list databases args)
$ sqoop-list-databases (generic args) (list databases args)
```

Syntax:

```
$ sqoop list-databases --connect
```

8. Sqoop-List-Tables

Used to list all the tables in a specified database. Generic Syntax:

```
$ sqoop list-tables (generic args) (list tables args)
$ sqoop-list-tables (generic args) (list tables args)
```

Syntax:

```
$ sqoop list-tables –connect
```

Conclusion:

Thus we installed Scoop successfully and studied the basic commands.

Experiment – 3

Aim - To install and configure MongoDB/ Cassandra/ HBase/ Hypertable to execute NoSQL commands.

Theory -

Hardware / Software Required: MongoDB

A NoSQL (originally referring to "non SQL" or "non-relational") database provides a mechanism for storage and retrieval of data which is modeled in means other than the tabular relations used in relational databases. Relational databases were not designed to cope with the scale and agility challenges that facemodern applications, nor were they built to take advantage of the commodity storage and processing power available today.

The Benefits of NoSQL

- When compared to relational databases, NoSQL databases are more scalable and provide superior performance, and their data model addresses several issues that the relational model is not designed to address:
- Large volumes of rapidly changing structured, semi-structured, and unstructured data.
- Agile sprints, quick schema iteration, and frequent code pushes.
- Object-oriented programming that is easy to use and flexible.
- Geographically distributed scale-out architecture instead of expensive, monolithic architecture.

MongoDB

It is an open-source document database, and leading NoSQL database. MongoDB is written in c++. It is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

MongoDB Installation steps (Ubuntu 18.04):

Step 1 — Update system

sudo apt-get update

Step 2 — Installing and Verifying MongoDB
Now we can install the MongoDB package itself.

sudo apt-get install -y mongodb
This command will install several packages containing latest stable version of MongoDB along with helpful management tools for the MongoDB server.

After package installation MongoDB will be automatically started. You can check this by running the following command.

sudo service mongod status
If MongoDB is running, you'll see an output like this (with a different process ID).

Output
mongod start/running, process 1611
You can also stop, start, and restart MongoDB using the service command (e.g. service mongod stop, service mongod start).

Commands
MongoDB ***use DATABASE_NAME*** is used to create database. The command will create a new database, if it doesn't exist otherwise it will return the existing database.

Syntax:
Basic syntax of **use DATABASE** statement is as follows:
use DATABASE_NAME

The dropDatabase() Method

MongoDBdb.dropDatabase() command is used to drop a existing database.

Syntax:

Basic syntax of dropDatabase() command is as follows:

db.dropDatabase()

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database

The createCollection() Method

MongoDBdb.createCollection(name, options) is used to create collection.

Syntax:

Basic syntax of createCollection() command is as follows

db.createCollection(name, options)

In the command, name is name of collection to be created. Options is a document and

used to specify configuration of collection

The find() Method

To query data from MongoDB collection, you need to use MongoDB'sfind() method.

Syntax

Basic syntax of find() method is as follows

>db.COLLECTION_NAME.find().

MongoDB'supdate() and save() methods are used to update document into a collection. The update() method update values in the existing document while the save() method replaces the existing document with the document passed in save() method.

MongoDBUpdate() method

The update() method updates values in the existing document.

Syntax

Basic syntax of update() method is as follows

>db.COLLECTION_NAME.update(SELECTIOIN_CRITERIA, UPDATED_DATA)

Experiment – 4

Aim - Experiment on Hadoop Map-Reduce / PySpark: -Implementing simple algorithms in Map-Reduce: Matrix multiplication, Aggregates, Joins, Sorting, Searching, etc.

Theory:

If M is a matrix with element m_{ij} in row i and column j, and N is a matrix with element n_{jk} in row j and column k, then the product:

$$P = M\,N$$

is the matrix P with element p_{ik} in row i and column k, where:

$$p_{ik} = \sum_{j} m_{ij}\, n_{jk}$$

We can think of a matrix M and N as a relation with three attributes: the row number, the column number, and the value in that row and column, i.e.,:

$$M\,(I, J, V) \text{ and } N\,(J, K, W)$$

with the following tuples, respectively:

$$(i, j, m_{ij}) \text{ and } (j, k, n_{jk}).$$

- In case of sparsity of M and N , this relational representation is very efficient in terms of space.
- The product M N is almost a natural join followed by grouping and aggregation. Analogously, send each matrix element n_{jk} to the key value pair:

$$(j, (N, k, n_{jk})) .$$

- Map: Send each matrix element m_{ij} to the key value pair:

$$(j, (M, i, m_{ij})) .$$

Analogously, send each matrix element n_{jk} to the key value pair:

$$(j, (N, k, n_{jk})) .$$

- Reduce: For each key j, examine its list of associated values. For each value that comes from M , say (M, i, m_{ij}), and each value that comes from N , say (N, k, n_{jk}), produce the tuple

$$(i, k, v = m_{ij}\, n_{jk}),$$

The output of the Reduce function is a key j paired with the list of all the tuples of this form that we get from j:

$$(j, [(i1, k1, v1), (i2, k2, v2), \ldots, (ip, kp, vp)]) .$$

- Map: Send each matrix element m_{ij} to the key value pair:

$$(j, (M, i, m_{ij})) .$$

Analogously, send each matrix element n_{jk} to the key value pair:

$$(j, (N, k, n_{jk})) .$$

- Reduce: For each key j, examine its list of associated values. For each value that comes from M , say (M, i, m_{ij}), and each value that comes from N , say (N, k, n_{jk}), produce the tuple

$$(i, k, v = m_{ij} n_{jk}),$$

The output of the Reduce function is a key j paired with the list of all the tuples of this form that we get from j:

$$(j, [(i1, k1, v1), (i2, k2, v2), \ldots, (ip, kp, vp)]) .$$

- Map: For each element m_{ij} of M , produce a key-value pair

$$((i, k), (M, j, m_{ij})) ,$$

for k = 1, 2, . . ., up to the number of columns of N .
Also, for each element n_{jk} of N , produce a key-value pair

$$((i, k), (N, j, n_{jk})) ,$$

for i = 1, 2, . . ., up to the number of rows of M .

Reduce: Each key (i; k) will have an associated list with all the values (M; j;mij) and (N; j; njk); for all possible values of j. We connect the two values on the list that have the same value of j, for each j:
We sort by j the values that begin with M and sort by j the values that begin with N, in separate lists, The jth values on each list must have their third components, mij and njk extracted and multiplied,
Then, these products are summed and the result is paired with (i; k) in the output of the Reduce function.

Conclusion – An algorithm in Map Reduce is implemented

Experiment – 5

Aim - Create HIVE Database and Descriptive analytics-basic statistics, visualization using Hive/PIG/R.

Theory:

What is Hive?

Apache Hive is a Data warehouse system which is built to work on Hadoop. It is used to querying and managing large datasets residing in distributed storage. Before becoming an open source project of Apache Hadoop, Hive was originated in Facebook. It provides a mechanism to project structure onto the data in Hadoop and to query that data using a SQL-like language called HiveQL (HQL).Hive is used because the tables in Hive are similar to tables in a relational database. Many users can simultaneously query the data using Hive-QL.

What is HQL?

Hive defines a simple SQL-like query language to querying and managing large datasets called Hive-QL (HQL). It's easy to use if you're familiar with SQL Language. Hive allows programmers who are familiar with the language to write the custom MapReduce framework to perform more sophisticated analysis.

Uses of Hive:

1. The Apache Hive distributed storage.

2. Hive provides tools to enable easy data extract/transform/load (ETL)

3. It provides the structure on a variety of data formats.

4. By using Hive, we can access files stored in Hadoop Distributed File System (HDFS is used to querying and managing large datasets residing in) or in other data storage systems such as Apache HBase.

Hive Commands :

Data Definition Language (DDL)

DDL statements are used to build and modify the tables and other objects in the database.

Go to Hive shell by giving the command sudo hive and enter the command _create database <data base name>' to create the new database in the Hive.

```
hive> create database retail;
OK
Time taken: 5.275 seconds
hive>
```

To list out the databases in Hive warehouse, enter the command _show databases‘.

```
hive> show databases;
OK
default
retail
Time taken: 0.228 seconds
hive>
```

The command to use the database is USE <data base name>

```
hive> use retail;
OK
Time taken: 0.023 seconds
hive>
```

Describe provides information about the schema of the table.

Data Manipulation Language (DML)

DML statements are used to retrieve, store, modify, delete, insert and update data in the database.

Example :

LOAD, INSERT Statements.

Syntax :

LOAD data <LOCAL> inpath <file path> into table [tablename]

The Load operation is used to move the data into corresponding Hive table. If the keyword local is specified, then in the load command will give the local file system path. If the keyword local is not specified we have to use the HDFS path of the file.

The insert command is used to load the data Hive table. Inserts can be done to a table or a partition.

• INSERT OVERWRITE is used to overwrite the existing data in the table or partition.

• INSERT INTO is used to append the data into existing data in a table. (Note: INSERT INTO syntax is work from the version 0.8)

CONCLUSION:
Hive is demonstrated successfully.

Experiment – 6

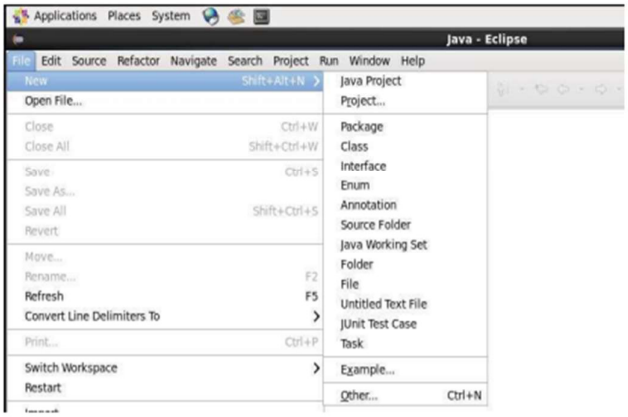
Aim - Write a program to implement word count programs using MapReduce.

Theory -

One of the three components of Hadoop is Map Reduce. The first component of Hadoop, that is, Hadoop Distributed File System (HDFS) is responsible for storing the file. The second component that is, Map Reduce is responsible for processing the file.
MapReduce also uses Java but it is very easy if you know the syntax on how to write it. It is the basis of MapReduce. So here are the steps which show how to write a MapReduce code for Word Count.

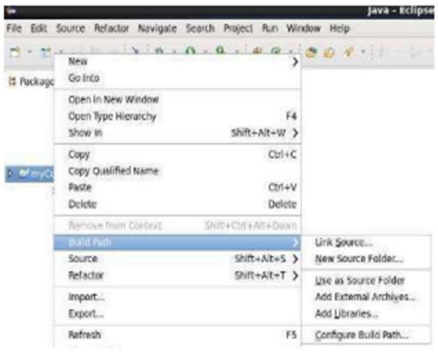
STEPS =>

1. Open Eclipse → then select File → New → Java Project → Name it WordCount → then Finish

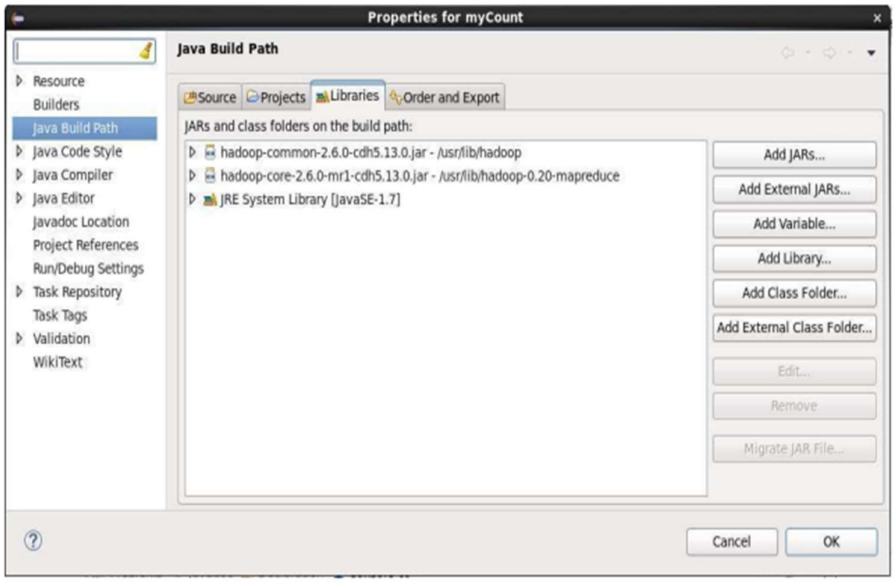


2. Create 3 Java Classes into the project. Name them WCDriver(having the main function), WCMapper, WCReducer.

3. You have to include two Reference Libraries for that: Right Click on Project -> then select Build Path-> Click on Configure Build Path
4. In the figure below, you can see the Add External JARs option on the Right Hand Side. Click on it and add the below mentioned files. You can find these files in /usr/lib/



a. /usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.13.0.jar
b. /usr/lib/hadoop/hadoop-common-2.6.0-cdh5.13.0.jar



Mapper Code:

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class WCMapper extends MapReduceBase implements Mapper<LongWritable,Text, Text,
IntWritable> {
        public void map(LongWritable key, Text value, OutputCollector<Text,IntWritable> output,
Reporter rep) throws IOException
        {
                String line = value.toString();
                for (String word : line.split(" ")){
                        if (word.length() > 0){
                                output.collect(new Text(word), new IntWritable(1));
                        }
                }
        }
}
```

Reducer Code:

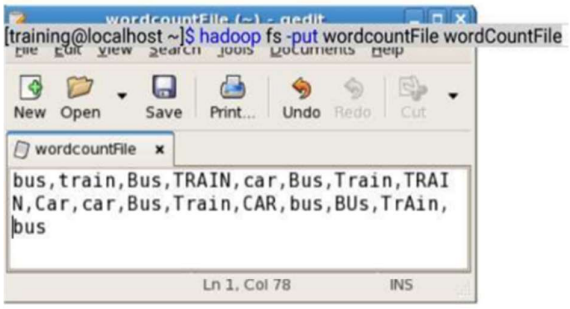
```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class WCReducer extends MapReduceBase implements Reducer<Text,IntWritable, Text,
IntWritable> {
        public void reduce(Text key, Iterator<IntWritable> value,OutputCollector<Text, IntWritable>
output,Reporter rep) throws IOException
        {
                int count = 0;
                while (value.hasNext()){
                        IntWritable i = value.next();
                        count += i.get();
                }
                output.collect(key, new IntWritable(count));
        }
}
```

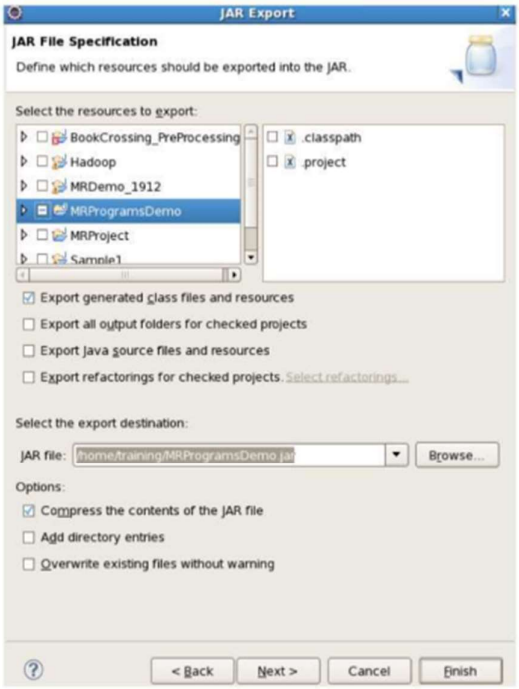
Driver Code:

```
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool {
        public int run(String args[]) throws IOException {
                if (args.length < 2) {
                        System.out.println("Please give valid inputs");
                        return -1;
```

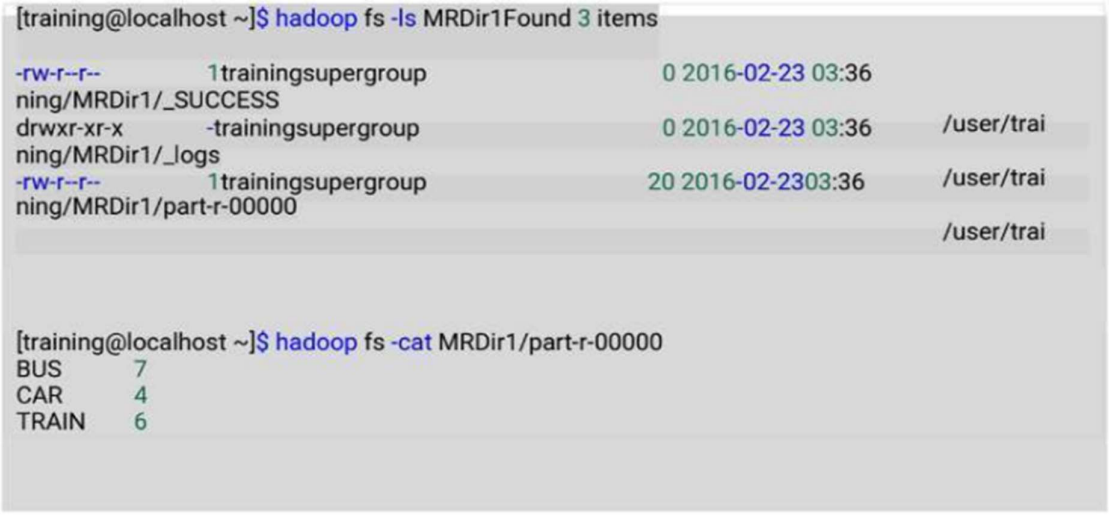
```
        }
        JobConf conf = new JobConf(WCDriver.class);
        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));
        conf.setMapperClass(WCMapper.class);
        conf.setReducerClass(WCReducer.class);
        conf.setMapOutputKeyClass(Text.class);
        conf.setMapOutputValueClass(IntWritable.class);
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        JobClient.runJob(conf);
        return 0;
    }
    public static void main(String args[]) throws Exception {
        int exitCode = ToolRunner.run(new WCDriver(), args);
        System.out.println(exitCode);
    }
}
```

5. Now you have to make a jar file. Right Click on Project-> Click on Export-> Select export destination as Jar File-> Name the jar File(WordCount.jar) -> Click on next -> at last Click on Finish. Now copy this file into the Workspace directory of Cloudera
6. Open the terminal on CDH and change the directory to the workspace. You can do this by using the ***"cd workspace/"*** command. Now, Create a text file(WCFile.txt) and move it to HDFS. Open the terminal and write this code(remember you should be in the same directory as the jar file you have created just now).
7. Now, run this command to copy the file input file into the HDFS "***hadoop fs -put WCFile.txt WCFile.txt"***





8. Run Jar File: (Hadoopjarjarfilename.jarpackageName.ClassNamePathToInputTextFile PathToOutputDirectory)
9. After Executing the code, you can see the result in the WCOutput file or by writing the following command on terminal. ***"hadoop fs -cat WCOutput/part-00000"***



CONCLUSION:

In this experiment, we have understood about the MapReduce in Hadoop and have also implemented the "WORD COUNT" program using it.

Experiment – 7

Aim - Implementing DGIM algorithm using any Programming Language/ Implement Bloom Filter using any programming language.

Bloom Filter is a data structure that can do this job.

For understanding bloom filters, you must know what is hashing. A hash function takes input and outputs a unique identifier of fixed length which is used for identification of input.

What is Bloom Filter?

A Bloom filter is a space-efficient probabilistic data structure that is used to test whether an element is a member of a set. For example, checking availability of username is set membership problem, where the set is the list of all registered username. The price we pay for efficiency is that it is probabilistic in nature that means, there might be some False Positive results.

False positive means, it might tell that given username is already taken but actually it's not.

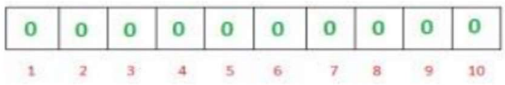
Interesting Properties of Bloom Filters

- Unlike a standard hash table, a Bloom filter of a fixed size can represent a set with an arbitrarily large number of elements.
- Adding an element never fails. However, the false positive rate increases steadily as elements are added until all bits in the filter are set to 1, at which point all queries yield a positive result.
- Bloom filters never generate **false negative** result, i.e., telling you that a username doesn't exist when it actually exists.

- Deleting elements from filter is not possible because, if we delete a single element by clearing bits at indices generated by k hash functions, it might cause deletion of few other elements. Example – if we delete "geeks" (in given example below) by clearingbit at 1, 4 and 7, we might end up deleting "nerd" also Because bit at index 4 becomes 0 and bloom filter claims that "nerd" is not present.

Working of Bloom Filter

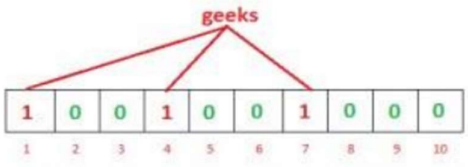
A empty bloom filter is a **bit array** of **m** bits, all set to zero, like this –



We need **k** number of **hash functions** to calculate the hashes for a given input. When we want to add an item in the filter, the bits at k indices h1(x), h2(x), … hk(x)are set, where indices are calculated using hash functions.
Example – Suppose we want to enter "geeks" in the filter, we are using 3 hash functions and a bit array of length 10, all set to 0 initially. First we'll calculate the hashes as following :

```
h1("geeks")   % 10 = 1
h2("geeks")   % 10 = 4
h3("geeks")   % 10 = 7
```

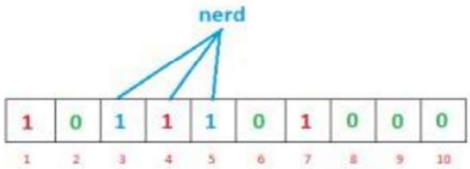
Note: These outputs are random for explanation only.
Now we will set the bits at indices 1, 4 and 7 to 1



Again we want to enter “nerd”, similarly we’ll calculate hashes

```
h1(“nerd”)    % 10 = 3
h2(“nerd”)    % 10 = 5
h3(“nerd”)    % 10 = 4
```

Set the bits at indices 3, 5 and 4 to 1



Now if we want to check “geeks” is present in filter or not. We’ll do the same process but this

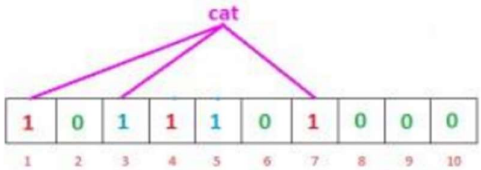
time in reverse order. We calculate respective hashes using h1, h2 and h3 and check if all these indices are set to 1 in the bit array. If all the bits are set thenwe can say that “geeks” is **probably present**. If any of the bit at these indices are 0 then “geeks” is **definitely not present**.

False Positive in Bloom Filters

The question is why we said **“probably present”**, why this uncertainty. Let’s understand this with an example. Suppose we want to check whether “cat” is presentor not. We’ll calculate hashes using h1, h2 and h3

```
h1(“cat”)    % 10 = 1
h2(“cat”)    % 10 = 3
h3(“cat”)    % 10 = 7
```

If we check the bit array, bits at these indices are set to 1 but we know that “cat” wasnever added to the filter. Bit at index 1 and 7 was set when we added “geeks” and bit3 was set we added “nerd”.



Experiment – 8

Aim - Implementing any one Clustering algorithm (K-Means/CURE) using Map-Reduce.

Theory - K-Means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters).The number of clusters should match the data. An incorrect choice of the number of clusters will invalidate the whole process. An empirical way to find the best number of clusters is to try K-means clustering with different number of clusters and measure the resulting sum of squares.

The basic K-means Algorithm is as follows:

Step 1: Select K points as initial centroids

Step2: repeat

Step 3: Form K clusters by assigning each point to its Closest centroid

Step 4: Recompute the centroid of each cluster

Step 5: Until centroids do not change.

The key step of Basic K-means algorithm is selection of proper initial centroids. Initial clusters are formed by random initialization of centroids.

Hadoop - Hadoop is a platform that provides both distributed storage and computational capabilities. It is an open source software project that enables the distributed processing of large data sets across clusters of commodity servers. It is designed to scale up from a single server to thousands of machines, with a very high degree of fault tolerance. Hadoop[7] is a distributed master-slave architecture that consists of Hadoop distributed file system (HDFS) for storage and Map-Reduce for computational capabilities. Hadoop can handle all types of data from disparate systems: structured, unstructured, log files, pictures, audio files, communications records, email – just about anything you can think of, regardless of its native format. Even when different types of data have been stored in unrelated systems, you can dump it all into your Hadoop cluster with no prior need for a schema.

Mapreduce K-Means Clustering - The mapreduce k-means clustering approach for processing big text corpus can be done by the following steps:

1) Give the Sequence file from directory of text documents as input.

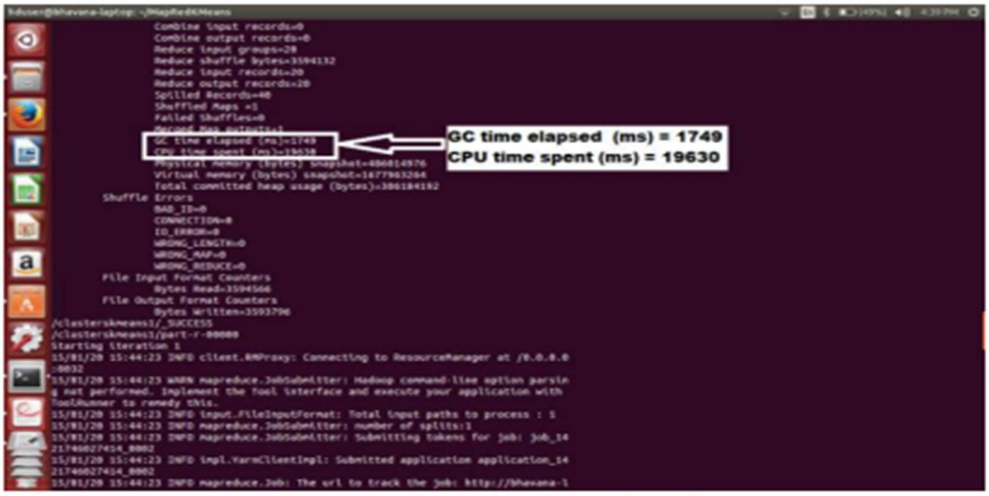
2) Tokenize and generate TF-IDF vector for each document from sequence file.

3) Apply Map-Reduce K-Means algorithm to form k clusters.

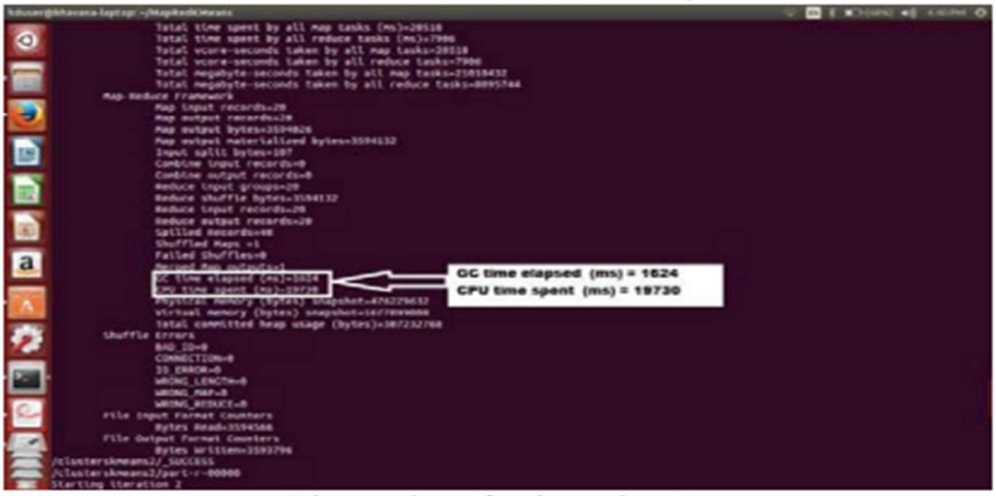
A. Sequence File from Directory of Text Documents Map reduce programming is coined to process huge data sets in parallel and distributed environment. Suppose, we select the input data from a document set, where the text files in the directory are small in size. Since HDFS and Mapreduce are optimized for large files, convert the small text files into larger file i.e., SequenceFile format. SequenceFile is a hadoop class, which allows us to write document data in terms of binary pairs, where key is a Text with unique document id and value is Text content within the document in UTF-8 format. SequenceFile packs the small files and process whole file as a record. Since the SequenceFile is in binary format, we could not able to read the content directly but faster for read /write operations. B. Creating TF-IDF Vectors The sequence file from the previous step is fed as Input to create vectors. The TF-IDF vectors are calculated in Mapreduce by the following steps: Step 1: Tokenization: The input fed to map function is in format of pairs, where key is the document name and value as document content. The outcome of reduce function is also pair where key is document name and value are tokens (words) present in that document. Ex: Key: /acq1.txt: Value: [macandrews, forbes, holdings, bids, revlon, mcandrews, forbes, holdings, inc,said, offer, dlrs, per, share, all, revlon, group] Step 2: Dictionary file: This step assign unique number to each token in all documents. The input format for the map function is and the output of reduce function is . <Word, uniqueid>.

MapReduce K-Means Algorithm The implementation of map reduce k-means accepts two input files. One input file contains the documents with each term and its tf-idf values, and the second is k initial centroids file. The set of k initial centroids are selected randomly. In every iteration, the map reduce framework splits the input data into M splits .

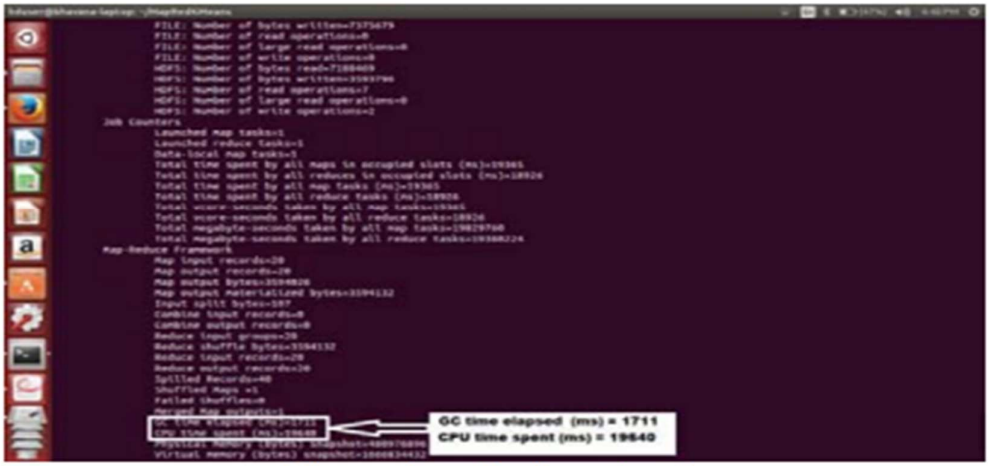
Output



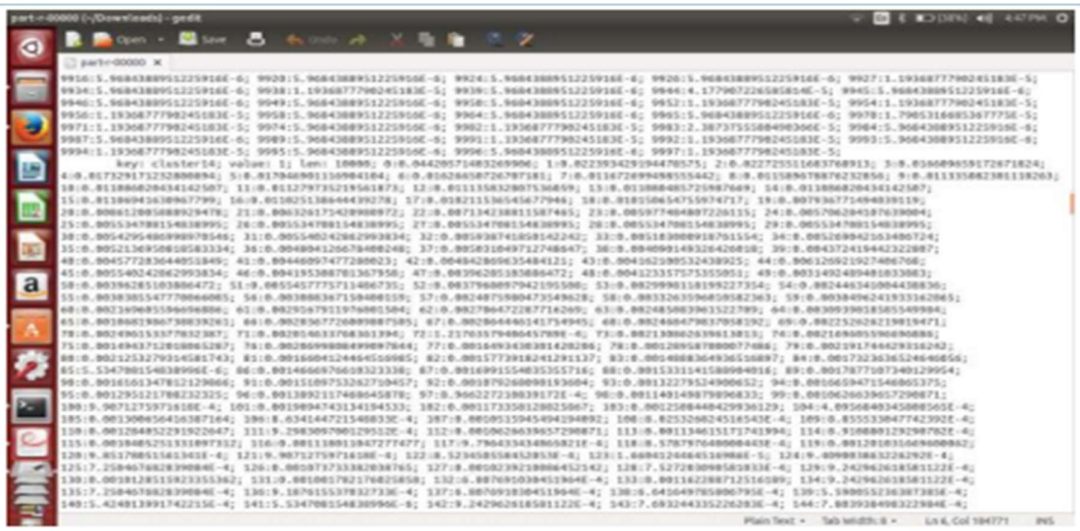
Time taken for iteration 0



Time taken for iteration 1



Time taken for iteration 2



Clusters

Conclusions – We had successfully implemented Clustering algorithm (K-Means) using Map-Reduce.

Experiment – 9

Aim - Streaming data analysis-use flume for data capture, HIVE/PYSpark for analysis of twitter data, chat data, weblog analysis etc.

Theory - In order to collect and process the streaming data from various streaming sites and produce an analytical report that helps to get a clear pictorial representation of events, the assets of streaming process generates massive volume of real time data mainly referred to the term “Big Data”. This system proposes to produce a graphical visualization (bar chart, pie-chart, etc.,) for sentimental analysis using NLP, analyzing particular event on particular period of time and WORD CLOUD in which each independent word acts as hyperlink. Those hyperlinks are being redirected to browser for geographical map with location tagging. The generated results will be useful for either government or private firms (News reports/Consultancies/Research). In order to aggregate, store and analyse the streaming data that are being generated Day-By-Day we get into the concept of Hadoop And Flume Technologies. Products of apache foundation been used as an open source tools for data analytics. With the help of these tools and technologies process of predicting and analyzing data has been made much easier and efficient. This proposed system designed by an API that helps to collect data from Twitter/ other streaming sites by using „#‟ tag/Keywords. Tweets by the News channel and retweets by the public are being collected. Those generated data serves as a input for analytical operation. Purpose of generating word cloud from the tweets which brings out the emotions of the country and tell the importance of the particular event along with the location by using the tool “Flume”. Tableau is connected to the cluster that helps in analysing and generating the word cloud.

Word clouds are one of the simplest and most intuitive ways of visualizing text data. The data collected of last two week, from most of all public tweets related to what the people are talking around the globe The word cloud are generated through the following process. First, a computer program takes a text and counts how frequent each word is., for example if there was a character named “Cat” as well as generic animals referred to as “cat”. Also remove stop words[1,2] , which add little to the final visualization in many cases. Second, creation of word frequency list.

Output

flume.conf

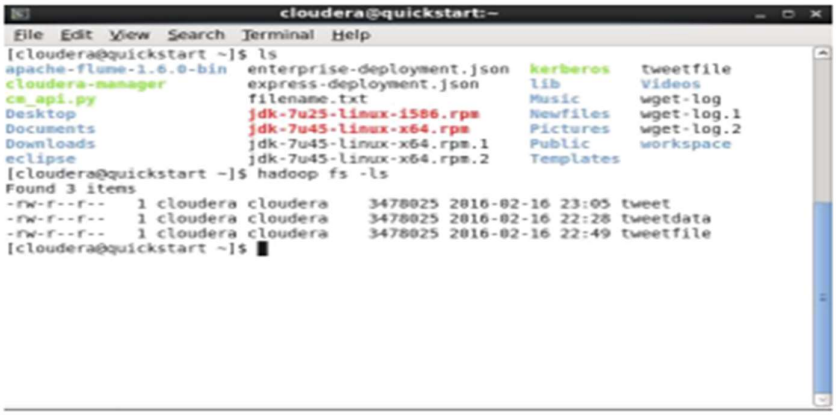
```
TwitterAgent.sources = Twitter
TwitterAgent.channels = MemChannel
TwitterAgent.sinks = HDFS

TwitterAgent.sources.Twitter.type = com.cloudera.flume.source.TwitterSource
TwitterAgent.sources.Twitter.channels = MemChannel
TwitterAgent.sources.Twitter.consumerKey = NpNLCC7eEf8ECmNQMcLTlU5Ay
TwitterAgent.sources.Twitter.consumerSecret = OX7hy8yAlpkjEIqJi3zJlR3loewIrM0eabmNLpCE7GvsN8Hvi0
TwitterAgent.sources.Twitter.accessToken = 250966561-Inbt4x2MNyhb0wQs9ULImIXsmN3Fm2fywPaKJsMf
TwitterAgent.sources.Twitter.accessTokenSecret = zZ2wG0896aAxHqg77QHBT21EbruQ1BjQ0eA9e43NPCu53

TwitterAgent.sources.Twitter.keywords = times of india, deccan chronicle, hindustan times, cnn ibn news, times now

TwitterAgent.sinks.HDFS.channel = MemChannel
TwitterAgent.sinks.HDFS.type = hdfs
TwitterAgent.sinks.HDFS.hdfs.path = hdfs://localhost:8020/user/flume/news/
TwitterAgent.sinks.HDFS.hdfs.fileType = DataStream
TwitterAgent.sinks.HDFS.hdfs.writeFormat = Text
TwitterAgent.sinks.HDFS.hdfs.batchSize = 1000
TwitterAgent.sinks.HDFS.hdfs.rollSize = 0
TwitterAgent.sinks.HDFS.hdfs.rollCount = 10000

TwitterAgent.channels.MemChannel.type = memory
TwitterAgent.channels.MemChannel.capacity = 10000
TwitterAgent.channels.MemChannel.transactionCapacity = 100
```

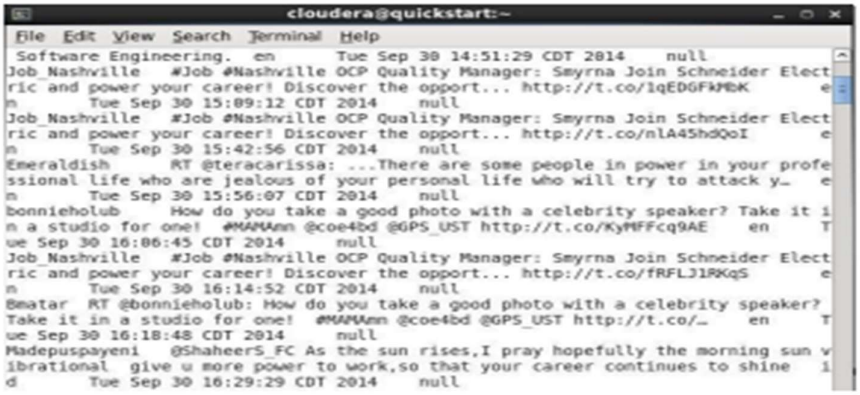
Browse Directory

/user/flume/news Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	cloudera	supergroup	2.37 KB	1	128 MB	FlumeData.1458019335411
-rw-r--r--	cloudera	supergroup	8.78 KB	1	128 MB	FlumeData.1458019372911
-rw-r--r--	cloudera	supergroup	7.63 KB	1	128 MB	FlumeData.1458019414286
-rw-r--r--	cloudera	supergroup	6.85 KB	1	128 MB	FlumeData.1458019459024

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ hadoop fs -ls /user/flume/news
Found 45 items
-rw-r--r--   1 cloudera supergroup       2424 2016-03-14 22:22 /user/flume/news/
FlumeData.1458019335411
-rw-r--r--   1 cloudera supergroup       8995 2016-03-14 22:23 /user/flume/news/
FlumeData.1458019372911
-rw-r--r--   1 cloudera supergroup       7809 2016-03-14 22:24 /user/flume/news/
FlumeData.1458019414286
-rw-r--r--   1 cloudera supergroup       7014 2016-03-14 22:24 /user/flume/news/
FlumeData.1458019459024
-rw-r--r--   1 cloudera supergroup      39023 2016-03-14 22:28 /user/flume/news/
FlumeData.1458019684487
-rw-r--r--   1 cloudera supergroup      24465 2016-03-14 22:29 /user/flume/news/
FlumeData.1458019718847
-rw-r--r--   1 cloudera supergroup      21602 2016-03-14 22:29 /user/flume/news/
FlumeData.1458019761086
-rw-r--r--   1 cloudera supergroup      27052 2016-03-14 22:30 /user/flume/news/
FlumeData.1458019794936
-rw-r--r--   1 cloudera supergroup      19069 2016-03-14 22:30 /user/flume/news/
FlumeData.1458019827859
-rw-r--r--   1 cloudera supergroup      16565 2016-03-14 22:31 /user/flume/news/
FlumeData.1458019859083
-rw-r--r--   1 cloudera supergroup      26565 2016-03-14 22:32 /user/flume/news/
FlumeData.1458019907981
```





<u>Conclusions</u> – We had successfully implemented streaming data analysis-use flume for data capture, HIVE/PYSpark for analysis of twitter data, chat data, weblog analysis etc.

Experiment – 10

Aim - Implementing Page Rank using Map-Reduce.

Theory - *PageRank* (or PR in short) is a recursive algorithm developed by Google founder *Larry Page* to assign a real number to each page in the Web so they can be ranked based on these scores, where the higher the score of a page, the more "important" it is.

To understand how these importance scores are being calculated, I want you to think of how would you decide if a person on Twitter is important or not? The first thing you would probably check is the number of his/her followers, where the more followers this person has, the higher likelihood he/she is famous(important), then you would maybe check how important this person's followers are, if the president for example is following him/her. The same idea is being applied in *PageRank* to find the importance score of a page, if a movie-page's link is being displayed on many other pages, we say that these pages are pointing or voting to that specif movie-page, and thus:

The importance of a page depends on the number of other pages pointing (votes) to it.

You might argue that if what we all do is counting the number of in-links for each page to find their importance, the clothes-seller in the early mentioned example, could simply fool the *PageRank* algorithm (searching engine) to believe that his page is important by creating a "spam farm" of a million pages, each of which linked to his clothes-page. However, as in Twitter, the president following a person gives more weight to the importance measurement than a normal follower, pages in the created "spam farm" would not be given much importance by *PageRank*, since other pages would not link to them, and the spammer will not be able to fool *Google*.

Output

In [1]:
```
import findspark
findspark.init()
findspark.find()
import pyspark
findspark.find()
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession
conf = pyspark.SparkConf().setAppName('appName').setMaster('local')
sc = pyspark.SparkContext(conf=conf)
spark = SparkSession(sc)
```

In [2]:
```
# Adjacency list
links = sc.textFile('links.txt')
links.collect()
```

Out[2]: ['A B C', 'B C D', 'C D', 'D A']

In [3]:
```
# Key/value pairs
links = links.map(lambda x: (x.split(' ')[0], x.split(' ')[1:]))
print(links.collect())

# Find node count
N = links.count()
print(N)

# Create and initialize the ranks
ranks = links.map(lambda node: (node[0],1.0/N))
print(ranks.collect())
```

```
[('A', ['B', 'C']), ('B', ['C', 'D']), ('C', ['D']), ('D', ['A'])]
4
[('A', 0.25), ('B', 0.25), ('C', 0.25), ('D', 0.25)]
```

In [3]:

```
ITERATIONS=20
for i in range(ITERATIONS):
    # Join graph info with rank info and propogate to all neighbors rank scores
    # And add up ranks from all in-coming edges
    ranks = links.join(ranks).flatMap(lambda x : [(i, float(x[1][1])/len(x[1][(
    .reduceByKey(lambda x,y: x+y)
    print(ranks.sortByKey().collect())
```

```
[('A', 0.25), ('B', 0.125), ('C', 0.25), ('D', 0.375)]
[('A', 0.375), ('B', 0.125), ('C', 0.1875), ('D', 0.3125)]
[('A', 0.3125), ('B', 0.1875), ('C', 0.25), ('D', 0.25)]
[('A', 0.25), ('B', 0.15625), ('C', 0.25), ('D', 0.34375)]
[('A', 0.34375), ('B', 0.125), ('C', 0.203125), ('D', 0.328125)]
[('A', 0.328125), ('B', 0.171875), ('C', 0.234375), ('D', 0.265625)]
[('A', 0.265625), ('B', 0.1640625), ('C', 0.25), ('D', 0.3203125)]
[('A', 0.3203125), ('B', 0.1328125), ('C', 0.21484375), ('D', 0.33203125)]
[('A', 0.33203125), ('B', 0.16015625), ('C', 0.2265625), ('D', 0.28125)]
[('A', 0.28125), ('B', 0.166015625), ('C', 0.24609375), ('D', 0.306640625)]
[('A', 0.306640625), ('B', 0.140625), ('C', 0.2236328125), ('D', 0.3291015625)]
[('A', 0.3291015625), ('B', 0.1533203125), ('C', 0.2236328125), ('D', 0.293945312
[('A', 0.2939453125), ('B', 0.16455078125), ('C', 0.2412109375), ('D', 0.30029296
[('A', 0.30029296875), ('B', 0.14697265625), ('C', 0.229248046875), ('D', 0.32348
[('A', 0.323486328125), ('B', 0.150146484375), ('C', 0.2236328125), ('D', 0.30273
[('A', 0.302734375), ('B', 0.1617431640625), ('C', 0.23681640625), ('D', 0.29870
[('A', 0.2987060546875), ('B', 0.1513671875), ('C', 0.23223876953125), ('D', 0.3
[('A', 0.31768798828125), ('B', 0.14935302734375), ('C', 0.22503662109375), ('D'
[('A', 0.30792236328125), ('B', 0.158843994140625), ('C', 0.2335205078125), ('D'
```

Conclusions – We had successfully implemented Page Rankusing Map-Reduce.

LTCE

AIM211
Pg.No.9)

Name :- Prathamesh S. Chilankar

ROLL NO. :- AIM211 BRANCH :- CSE - (AIML)

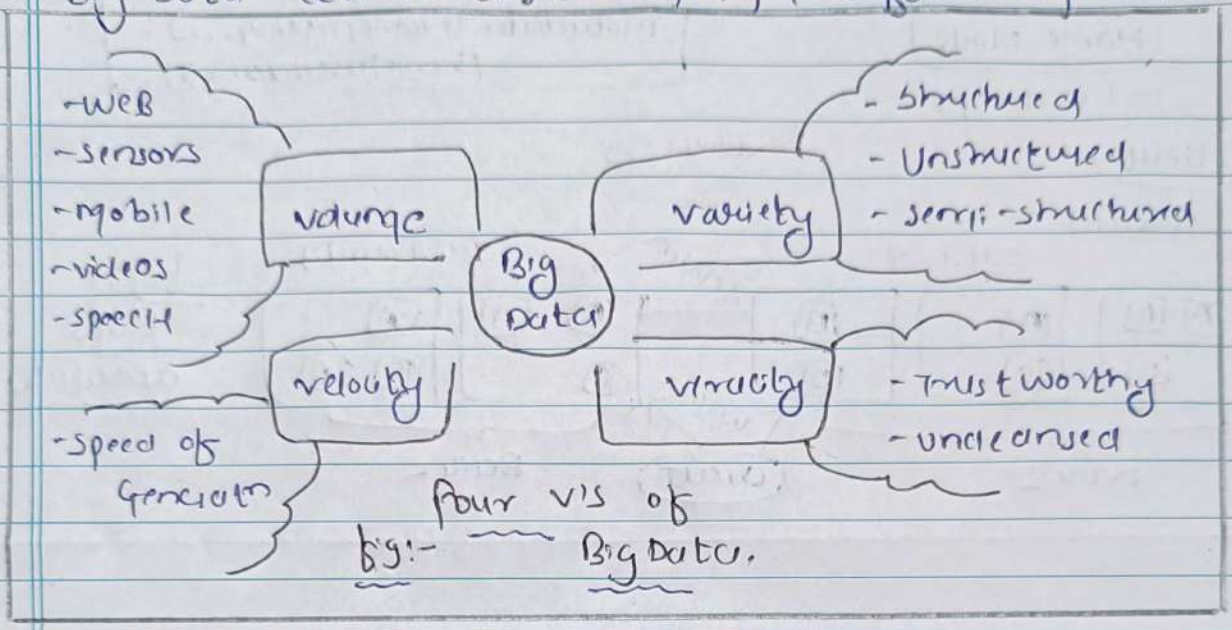
Subject :- Big Data Analytics (BDA)

Year :- BE Topic :- Assignment No. 01

SIGN :- Prathy Date :- August '23

Q. What are the characteristics of Big Data?

Big Data can be described by the following characteristics:



Four V's of Big:- Big Data.

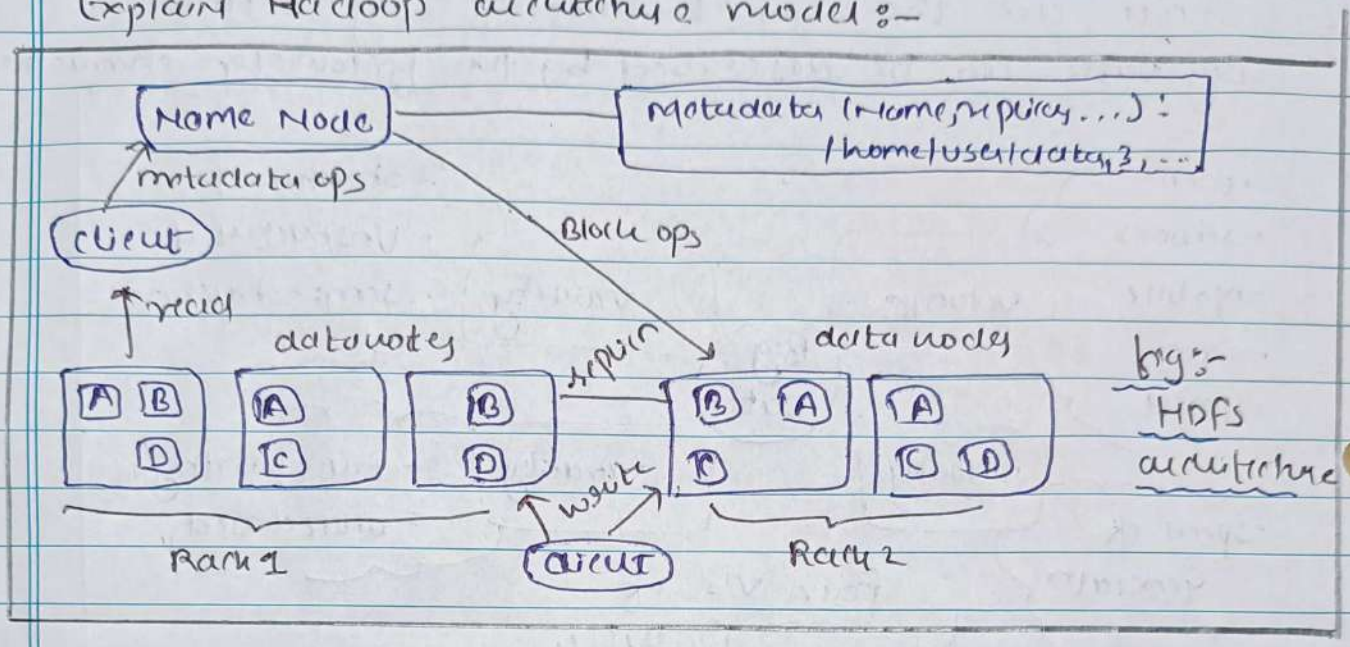
* **Variety :-**
Big Data often comes from a wide variety of sources. Collating this data is large-consuming, costly and technically challenging

* **Volume :-**
The sheer size of data makes it difficult to maintain & harder still to process into useful information.

* **Velocity :-**
Big data is collected at high speed. The company needs to quickly integrate & analyze it & new data to make best use of it.

* **Veracity :-**
Big data only as useful when it is reliable, however working a big dataset for reliability & accuracy difficult & large-consuming

Q. Explain Hadoop architecture model :-



> HDFS is a distributed file system that provides a limited interface for managing file system to allow it to scale and provide high throughput.

> Main components of HDFS :

(1) Name Node :

Name Node is the master that contains metadata.

a) manages namespace of file system in memory.

b) maps inode to list of block & location.

(2) DataNodes :

Datanodes are slaves which provide actual storage & are deployed on each machine.

a) Handles block storage on multiple volumes & also maintains block integrity.

b) periodically sends heartbeats & also block reports to the Namenode.

Q. Explain matrix multiplication by map Reduce.

> Technique in which a huge program is subdivided into small tasks and run parallely to make computation fastly, save time and mostly used in DS. (distributed systems)

> Example: $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$

> one step matrix multiplication has 1 mapper & 1 reducer.

> mapper for matrix A (K,V) = ((i,k)(A,j,Aij)) for all k
> B (K,V) = ((i,k)(B,j,Bjk)) for all i

matrix	i/p i|k	o/p (i,j)	value	(matrix, j, value).
A	0	0	1	(A, 0, 1)
A	0	1	2	(A, 1, 2)
A	1	0	3	(A, 0, 3)
A	1	1	4	(A, 1, 4)
B	0	0	5	(B, 0, 5)
B	0	1	6	(B, 0, 6)
B	1	0	7	(B, 1, 7)
B	1	1	8	(B, 1, 8)

> i|k | Aij x Bjk

i	k			
0	0	(A,0,1) (A,1,2) (B,0,5) (B,1,7)	(1x5 + 2x7)	5 + 14 = 19
0	1	(A,0,1) (A,1,2) (B,0,6) (B,1,8)	(1x6 + 2x8)	6 + 16 = 22
1	0	(A,0,3) (A,1,4) (B,0,5) (B,1,7)	(3x5 + 4x7)	28 + 15 = 43
1	1	(A,0,3) (A,1,4) (B,0,6) (B,1,8)	(3x6 + 4x8)	18 + 32 = 50

> Ans:- matrix : $\begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$ //

Q. List Relational Algebra operation

> duplicate rows are implicitly eliminated.

> operation:

1. Selection
2. projection
3. union
4. intersection
5. Natural join
6. Grouping & aggregating using map reduce.

Explain any two operation:

* Selection:- Select (where clause in SQL) lets you apply common condition over data you have & only get rows that satisfy the condn.

example:

Name	Age
John	21
Tom	17
mike	22
smith	16
Don	25

Selection (Age>20) →

Name	Age
John	21
mike	22
Don	25

* Grouping & aggregating :- Groups rows based on some set of columns and apply some aggregation (sum, count, max, min, etc..) on some column of small group that are formed. (this corresponds to groupby in SQL).

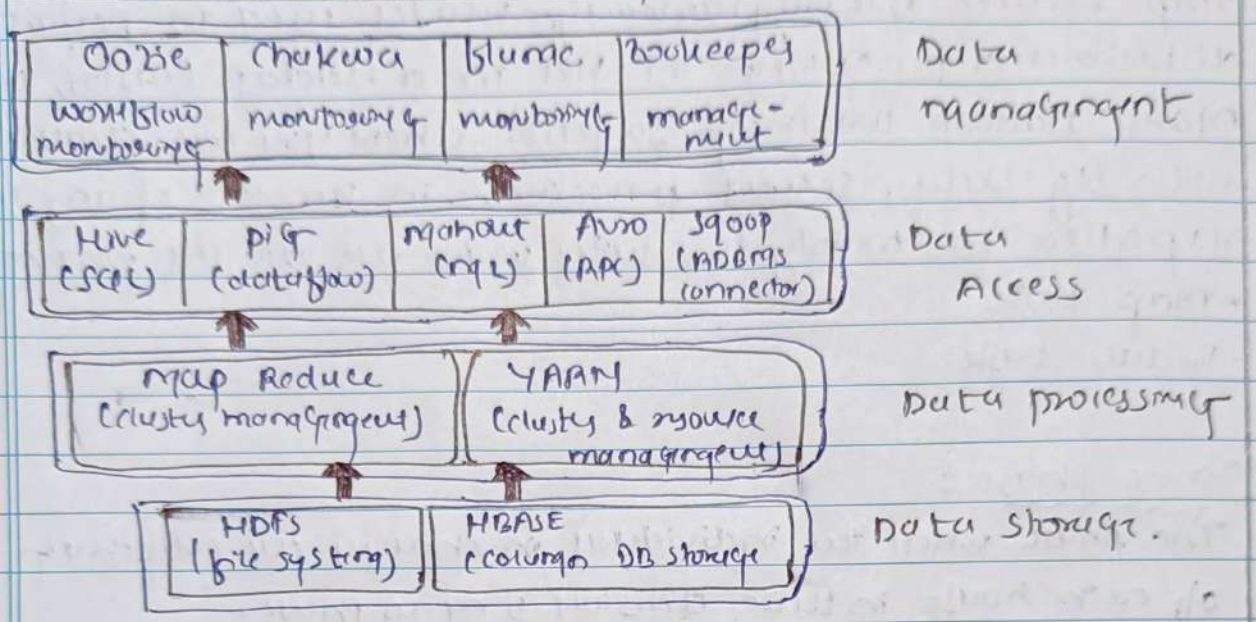
example :-

Name	winning
John	200
Tom	100
smith	500
John	100
Tom	400

Groupby (Name) Aggr (sum(winning)) →

Name	winning
John	300
Tom	500
smith	500

Q. Explain Hadoop Ecosystem



Eg:- Hadoop Ecosystem ..

> Components:

1) HDFS: (Hadoop distributed file system)
Stores & manage data in distributed manner eg:- NTFS.
2) YARN: (Yet another resource manager):
Cluster resource manager eg:- OS
3) map reduce: data processing eg:- CPU
4) Hadoop common: provide necessary libraries..

> - flume: data collector (unstructured, streaming DB, raw) in HDFS
- sqoop: data collector (structured like SQL, DB) in HDFS
- HBase: columnar store (nosql) to store large amount of data
- Hive: SQL query (apply preprocessing of data to analyze)
- pig (scripting): reduce no. of JAVA code.
- mahout (ML): ML algo. are there for scalability)
- oozie: for scheduling (manage, monitor data).
- zookeeper: coordinator → manage all tools for high clusty.

Q. Explain in detail map Reduce programming model.

> map reduce is a programming model used to perform distributed processing in it in a Hadoop cluster, which makes Hadoop working so fast. when you are dealing with Big Data, serial processing is no more of any use.

> mapreduce has mainly two tasks which are divided phase-wise:
- Map task
- Reduce task

> ① Map phase:
The phase where the individual in-charges are collecting population of each house in their division is map phase.
- mappers: Involved individual in charge for calculating population.
- i/p splits: the state or the division of the state.
- key-value pair: o/p form each individual mapper where key is Rajastan and value is 2.

> ② Reduce phase: The phase where you are aggregating result.
- Reducers: individuals who are aggregating the actual result. Here in our let's say trained-officers. Each reducer produce the o/p as a key-value pair.

> ③ Shuffle phase:-
The phase where the data is copied from mappers to Reducers is shuffler's phase.
It comes in b/w map and Reduce phase.
Now the map phase, Reduce phase, and Shuffle phase our the three main phases of our mapreduce.

Q. Explain in detail what is Big Data.

> Big Data refers to the massive datasets that are collected from a variety of data sources for business needs to reveal new insights for optimized decision making.

> Every day around 10 million text messages are sent, facebook has millions of active accounts and friends share content, photos and videos.

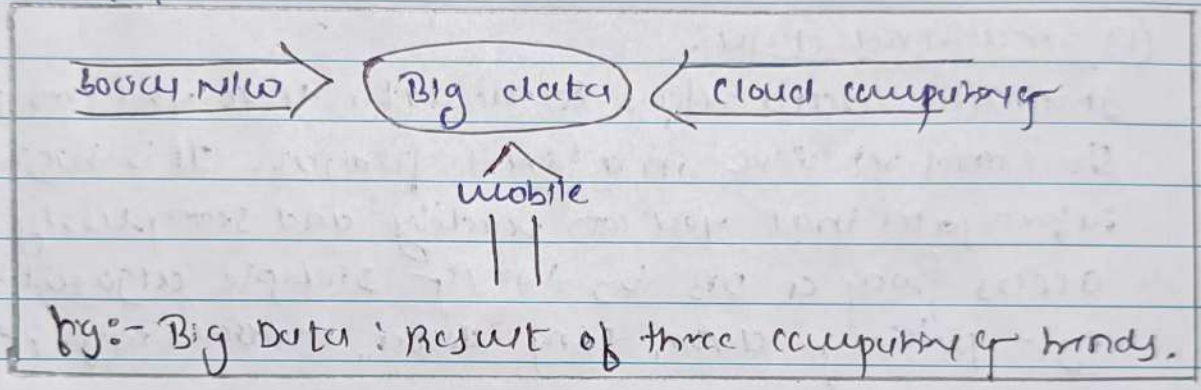


fig:- Big Data : Result of three computing trends.

> Big data analytics as shown in above fig is the result of three major trends in computing : mobile computing using hand-held devices, such as smartphones and tablets ; Social Nlw, such as facebook & Pinterest ; and cloud computing by which one can rent & have h/w setup for storing & computing.

> Challenges & considerations :
- Storage
- Processing
- Analysis
- privacy & security

> Opportunities & Impact :
- Healthcare
- Transportation
- Banking & financial services
- Smart city
- Business insights.

Q. Explain Types of Big Data.

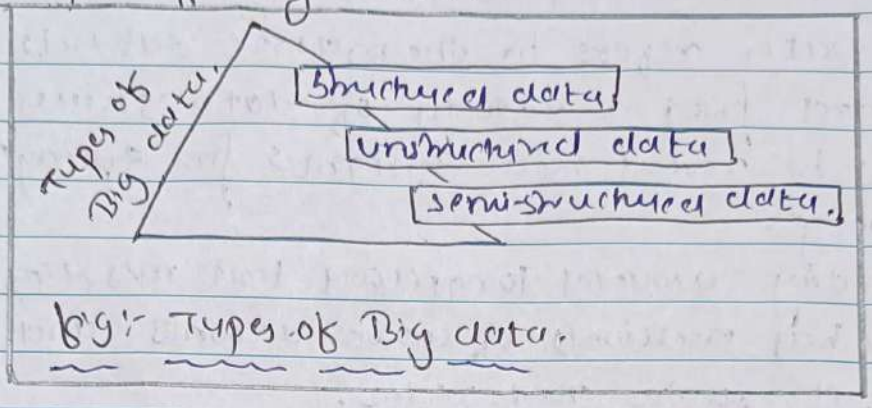


fig:- Types of Big data.

> ① Structured data:
Structured data refers to the data that you can process, store and retrieve in a fixed format. It is highly organized information that you can readily and seamlessly store & access from a DB by using simple algorithms.
eg:- pricing data, CRM data, medical info., etc..

> ② Unstructured data:
Unstructured data is the kind of data that allows adhere to any definite schema or set of rules. Its arrangement is unplanned and haphazard.
eg:- video files, Emails, images.

> ③ semi-structured data:
Semi-structured data pertains the data containing both formats that is structured and unstructured data.
Semi-structured data is a type of structured data that doesn't hold to the tabular structure of data, rarely associated with real data types of data but it includes tags or other markers to separate semantic elements.
eg:- HTML, web pages, Emails are eg. of semi-structured data.

Name :- Prathamesh S. Chikankar

Roll No. :- AIML11 Branch :- CSE - (AI&ML)

Year :- BE Subject :- Big Data Analytics (BDA)

Topic :- Assignment No. 02

Sign :- Prathu Date :- October'23

Q.1. Compare SQL and NoSQL

	SQL	NoSQL
1)	Relational DB management system [RDBMS]	Non-relational or distributed database system
2)	These database have fixed or static or predefined schema	They have dynamic schema.
3)	Best suited for complex queries	Not suited for complex queries
4)	vertically scalable	Horizontally scalable
5)	follows ACID property	follows CAP property.
6)	consist of Rigid schema	Consist of flexible schema
7)	Eg:- Oracle, MySQL	Eg:- mongoDB & couchDB

Q.2. Explain NoSQL Database stores.

NoSQL Database were born out of rigidity of traditional relational or SQL databases, which use tables, columns, and rows to establish relationship across data.

There are 4 Data architecture patterns in NoSQL data

> ① Key-value stores:

One of the most basic NoSQL database model is key-value. The data is collected in pattern of key-value pairs; A series of string, integers or characters is typically the key, but it can also be more advanced.

eg:- Dynoma DB [developed by amazon]
Berkley DB [developed by oracle]

> ② column database: (datastores)

This pattern employs data storage in divided cells that is further divided into columns. related than storing data in

PTO

relationship types. DB that are column.
eg:- Hbase, cassandra.

> (3) Document database store:
In the form of key-value pairs, the record database field fetches & accompanies info. but here the values are called documents. It is hierarchical version of key-value DB.
eg:- MongoDB, COUCHDB.

> (4) Graph Database store:
This patterns of architecture clearly deals with Mongodb storage and management in graphs. It essentially represents relation btw two or more objects.
eg:- Orient DB, flock DB.

Q.3. Agility is a NoSQL business drive. Justify.
> Agility is ability to accept change easily & quickly.
> Among the variety of agility dimensions such as model agility, operational agility & programming agility - one of the most important is to quickly & seamlessly scale on operation to accomodate large data.
> The most complex part of building application using RDBMS is the process of putting data into & getting data out of the database.
> The responsibility thus to generate correct combination of layers.
> All this hurdles are best overcome by NoSQL database.
> They can accomodate application changes easily & handle any volume of data effectively.
> Hence, this agility has become business drive for NoSQL DB.

Q.4. Explain the role & effect of damping factor in BDA:

Role:- (damping factor)

① Regularization: technique used in ML and statistical modelling to prevent overfitting, where model become too complex & fit training data too closely... L1 (lasso) & L2 (ridge)

② control of algorithm: in BDA, damping effect can be analogously compared controlling behaviour of algo. & goals.
eg:- feature selection, dimensionality reduction, outlier detection.
The damping factor is parameter introduced in page rank & other iterative to stimulate behaviour of user.

Effects:- (damping factor)

① stabilizing iteration: A damping factor helps in stabilizing the iterative process. without it, some webpages might end up with extremely high ranks, leading to unstable results.

② Balancing- & Global influence: Balance the influences of local links (link with a pages content) & Global

③ convergence & efficiency: Improve the efficiency & convergence of optimization process. This is turn lead to faster model training & analysis, making it more practical. for handling large scale data.

Q.5. List the different Nosql data stores.

- key-value store
- Document store
- Graph store
- Column store.

Explain any two of them.

7 ① key-value store:

Most basic Nosql database model. The data collected in

PTO

pattern of key-value pair, as the name implies. The key-value is a pair store in hash value which is unique.

eg:- Key-value

	key	value
Key-value	user-123	"John-Doe"
Key-value	file-123.pdf	<pdf document>
Key-value		

> (2) Document store:

In key-value pair, the record database features & corresponds also. but the value are called documents. The complicated data structure can be represent as text.. (form of arrays, etc.)

eg:-

key	Document
document-1	{ "Id" : "1" "name" : "sudham" "isactive" : "true" }
document-2	{ "Id" : "2" "name" : "prathamesh" "isactive" : "false" }

Q. 6. Explain FM algorithm with example.

> Flajolet-Martin algorithm approximate the no. of unique objects in a stream or a database in one pass.

> If the stream contains one element with an m of them uniques this algorithm runs in O(m) time & need O(log m) memory.

> Space-consumption is less in this algorithm

Example: determine distinct element in string using FM algo.

- i/p → 1, 3, 2, 1, 2, 3, 4, 3, 1, 2, 3, 1
- hash fn, h(x) → 6x+1 mod 5

PTO

- ① calculating hash fn

$h(x) = 6x+1 \bmod 5$

$h(1) = h(1)+1 \bmod 5$

$= 7 \bmod 5 = 2$

$h(1) = 2$
$h(2) = 3$
$h(3) = 4$
$h(4) = 0$

- ② Binary hash fn

$h(1) = 2 = 010$
$h(3) = 4 = 100$
$h(2) = 3 = 011$
$h(4) = 0 = 000$

- ③ trailing zeros:

$h(1) = 2 = 010 = 1$
$h(2) = 3 = 011 = 0$
$h(3) = 4 = 100 = 2$
$h(4) = 0 = 000 = 0$

- ④ maximum no.

the value of $r = 2$, distinct value, $R = 2^r \Rightarrow 2^2 \Rightarrow 4$ //

Ans:- hence, here are 4 distinct elements, 1, 2, 3, 4

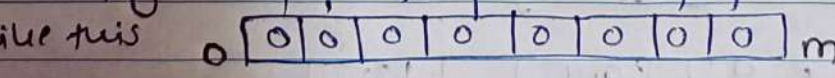
Q. 7 write a detail note on Bloom filter.

) A bloom filter is a space-efficient probabilistic DS that is used to tyt whether an element is member of set.

) eg:- checking availability of username in a set membership probing, where set is list of all registered username.

) only probing with Bloom filter is that they are probabilistic in nature that means there might be some FP results.

) eg:- empty bloom filter of bit array of m bits all set to 0, like this



- Suppose, we want to enter throw in filter, calculating hash fn.

$h(1)$ ("throw") $\%10 = 1$
$h(2)$ ("throw") $\%10 = 4$
$h(3)$ ("throw") $\%10 = 7$

PTO

- now, we will set bits of index 1, 4 & 7 to 1.

0	1	2	3	4	5	6	7	8	9
0	1	0	0	1	0	0	1	0	0

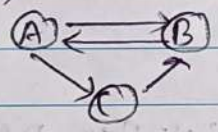
- again, we want to enty "catch" in the filty
- hashing: h(1) ("catch") %10 = 3
  h(2) ("catch") %10 = 5
  h(3) ("catch") %10 = 4

0	1	2	3	4	5	6	7	8	9
0	1	0	1	1	1	0	1	0	0

↓ 2

- as we observe in 4th position it appears as 2, it will gives FP, value as it is clearly exist, but comes first hence as value of "catch"

Q.8. compute simplified page Rank using damping factor.

(a) 

$$\rightarrow M = \begin{bmatrix} A & 0 & 1/2 & 1/2 \\ B & 1 & 0 & 0 \\ C & 0 & 1 & 0 \end{bmatrix}$$

damping factor = 0.9

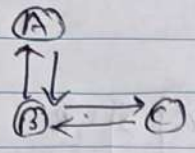
→ transpose → $M^T = \begin{bmatrix} 0 & 1 & 0 \\ 1/2 & 0 & 1 \\ 1/2 & 0 & 0 \end{bmatrix}$. M^T * damping factor

$$M^1 = \begin{bmatrix} 0 & 1 & 0 \\ 1/2 & 0 & 1 \\ 1/2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.9 \\ 0.9 \\ 0.9 \end{bmatrix} = \begin{bmatrix} 0.9 \\ 1.35 \\ 0.45 \end{bmatrix}$$

$$M^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1/2 & 0 & 1 \\ 1/2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.9 \\ 1.35 \\ 0.45 \end{bmatrix} = \begin{bmatrix} 1.35 \\ 0.9 \\ 0.45 \end{bmatrix}$$

page rank after two iteration, $r = \begin{bmatrix} 1.35 \\ 0.9 \\ 0.45 \end{bmatrix}$

PTO

(b) 

$$\rightarrow M = \begin{bmatrix} A & 0 & 1/2 & 0 \\ B & 1 & 0 & 1 \\ C & 1 & 1/2 & 0 \end{bmatrix}$$

damping factor = 0.8

$$\text{Transpose} = M^T = \begin{bmatrix} 0 & 1 & 1 \\ 1/2 & 0 & 1/2 \\ 0 & 1 & 0 \end{bmatrix}$$

M^T * damping factor

$$M^1 = \begin{bmatrix} 0 & 1 & 1 \\ 1/2 & 0 & 1/2 \\ 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0.8 \\ 0.8 \\ 0.8 \end{bmatrix} = \begin{bmatrix} 1.6 \\ 0.8 \\ 0.8 \end{bmatrix}$$

$$M^2 = \begin{bmatrix} 0 & 1 & 1 \\ 1/2 & 0 & 1/2 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1.6 \\ 0.8 \\ 0.8 \end{bmatrix} = \begin{bmatrix} 1.6 \\ 1.2 \\ 0.8 \end{bmatrix}$$

Final page rank after two iteration, $r = \begin{bmatrix} 1.6 \\ 1.2 \\ 0.8 \end{bmatrix}$

Q. 9. Explain CURE algorithm.

> Clustering is using for discovering groups & identifying interesting distributions in underlying data.

> Traditional clustering algorithm either favour clusters with spherical shapes & similar size, or are very fragile in presence of outliers

> more robust to outliers & identifies clusters having non-spherical shape & wide variance in size.

> CURE stands for clustering - using Representatives.

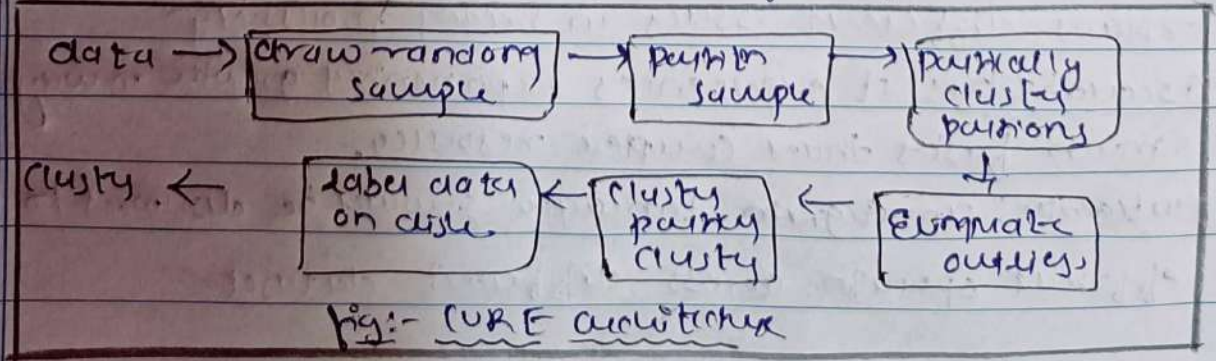
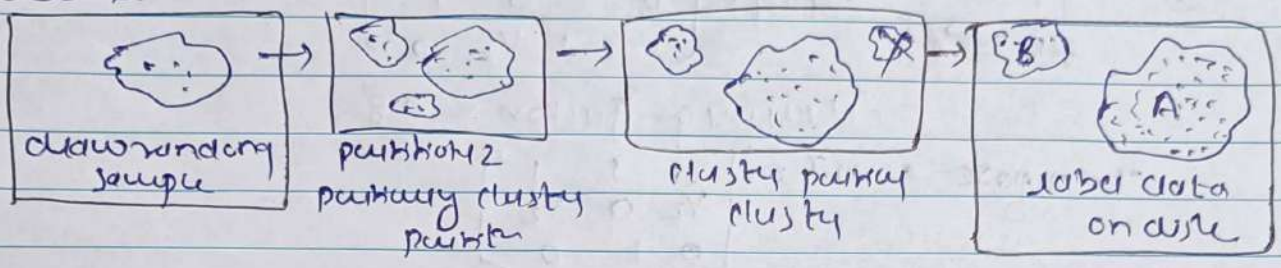


fig:- CURE architecture

PTO

example:



Q.10. Explain DGIM algorithm. Give the updating bucket approach.

> DGIM algorithm → Datar-Gion-is-Indyk-motwani algorithm
> designed to find number of 1s in a dataset
> uses $O(\log^2 N)$ bits to represent with an error of 50% more
> components of DGIM algorithm: Timestamp, Buckets.

* Rules for DGIM algorithm..

① Right side of stream always starts with 1.
eg:- 1001011 → bucket of size 4, having four 1's & starting with 1
② Every bucket should have atleast one 1, or no bucket formed.
③ All bucket should be in power of 2.
④ Bucket cannot decrease in size of more left (more in increasing order to left).

* eg:- ....10101100010111011001010...
N = 24 [window size]

..[10101]1[0000][10111]0[11]00[101][1]0...
$2^2=4$ $2^2=4$ $2^1=2$ $2^1=2$ $2^0=1$ ← size of bucket

Q.11. Explain different issues in stream processing

① Scalability: It represents exponential growth in way much faster than compute resources.
② Integration: an integration technique should be designed to enable efficient operation across different dataset

PTO

③ fault tolerance : high fault tolerance is required
④ Timeless : main challenge implementing distributed algorithms that will aggregate local value info. of data into global view with minimum latency b/w communicating nodes
⑤ achieving high consistency : non-trivial as it is difficult to determine when data are needed & when work should be consistent.

Q. 12. Compare DBMS & DSMS

	DBMS	DSMS
1)	Database management system	Data sharing management system
2)	Deals with persistent data	deals with sharing data.
3)	Random access data takes place	Sequential data access takes place
4)	Based on Query driven processing model	Based on data driven processing model.
5)	Data update rate is low	data update rate is high
6)	Uses unbounded disk storage	Uses main memory storage bounded
7)	Normal huge service	provides real huge service in DSMS.

Q. 13. Explain distance measure by suitable example.

a) Euclidean distance : length of line b/w two points (pythagoras thm)

$$d(x,y) = \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$$

eg:- x & y : (5,1), (9,-2) $= \sqrt{(9-5)^2 + (-2-1)^2} \Rightarrow \sqrt{25} \Rightarrow 5$

b) Jaccard distance : measure of dissimilarity b/w two datasets and is calculated by, $JD = 1 - SIM\ d(x,y)$

eg:- A = 0, 1, 2, 3, 4, 5
B = 6, 7, 8, 9, 10

similarity $J(A,B) = \left| \frac{A \cap B}{A \cup B} \right|$

$A \cap B = 0$, $A \cup B = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$

Jaccard distance $= 0/11 = 0 \Rightarrow 1 - 0 \Rightarrow 1$

PTO

> c) cosine distance: $\frac{A \cdot B}{\|A\| \|B\|}$ | eg:- A=1,2,-1 & B=2,1,1

$A \cdot B = 1 \times 2 + 2 \times 1 + (-1) \times 1 = 3$

$= 3/\sqrt{6} \cdot \sqrt{6} = 1/2$ $\|A\| \|B\| = \sqrt{6} \cdot \sqrt{6} = 6$

> d) Edit distance: distance blw two strings smallest no. of inserts & delete.

> e) Hamming distance: to perform XOR open to calculate hamming

eg:- 11011001 ⊕ 10011101 ⇒ 01000100.

SMA, it contains 2 1's, hence hamming distance = 2..

Q.14. Explain CAP theorem, Justify how it's diff from ACID properties.

> CAP theorem also called as brewers theorem

> states that it's impossible for distributed data store to offer more than two at of three guarantees

(1) consistency: data remains consistent after execution of operation

(2) Availability: system is always on, no single down.

(3) partition tolerance: the system continues to function even the communication among servers is unreliable..

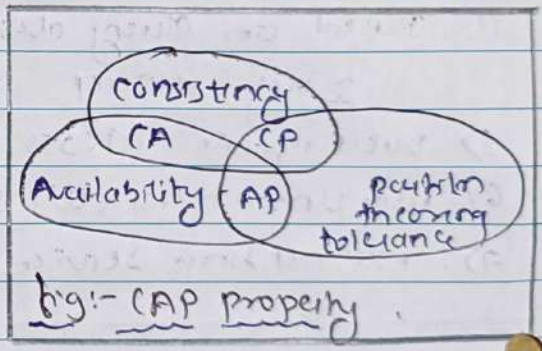


fig:- CAP property

> The uses of word consistency in CAP & its use in ACID do not refer to same identical concept.

> In CAP theorem, consistency refers to consistency of the values in different copies of same data item in a replicated distributed system. In ACID, it refers to the fact that transaction will not violate integrity constraints specified on the database schema.