METHODIST
COLLEGE OF ENGINEERING & TECHNOLOGY
(An UGC-AUTONOMOUS INSTITUTION)
Accredited by NAAC with A+ and NBA
Estd : 2008 Affliated to Osmania University & Approved by AICTE

LABORATORY MANUAL

BIG DATA ANALYTICS LAB

BE VI Semester (AICTE Model Curriculum): 2022-23

NAME: K KEERTHI_____

ROLL NO:_____

BRANCH:_____AI&DS--- SEM:_____VI____

DEPARTMENT OF COMPUTER SCIENCE &ENGINEERNG

Empower youth- Architects of Future World

VISION

To produce ethical, socially conscious and innovative professionals who would contribute to sustainable technological development of the society.

MISSION

To impart quality engineering education with latest technological developments and interdisciplinary skills to make students succeed in professional practice.

To encourage research culture among faculty and students by establishing state of art laboratories and exposing them to modern industrial and organizational practices.

To inculcate humane qualities like environmental consciousness, leadership, social values, professional ethics and engage in independent and lifelong learning for sustainable contribution to the society.

DEPARTMENT

OF

COMPUTER SCIENCE AND ENGINEERING(AI&DS)

LABORATORY MANUAL
BIG DATA ANALYTICS LAB

PreparedBy
K KEERTHI
Assistant Professor

VISION & MISSION

VISION

To become a leader in providing Computer Science & Engineeringeducation with emphasis on knowledge and innovation.

MISSION

- To offer flexible programs of study with collaborations to suit industryneeds.
- To provide quality education and training through novel pedagogicalpractices.
- To expedite high performance of excellence in teaching, research andinnovations.
- To impart moral, ethical values and education with social responsibility.

PROGRAM EDUCATIONAL OBJECTIVES

After 3-5 years of graduation, the graduates will be able to

PEO1: Apply technical concepts, Analyze, synthesize data to Design and create novel products and solutions for the real life problems.

PEO2: Apply the knowledge of Computer Science Engineering to pursue higher education with due consideration to environment and society.

PEO3: Promote collaborative learning and spirit of team work through multidisciplinary projects

PEO4: Engage in life-long learning and develop entrepreneurial skills.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM OUTCOMES

Engineering graduates will be able to:

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the Engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

At the end of 4 years, Computer Science and Engineering graduates at MCET will beable to:

PSO1: Apply the knowledge of Computer Science and Engineering in various domains like networking and data mining to manage projects in multidisciplinary environments.

PSO2: Develop software applications with open-ended programming environments**.**

PSO3: Design and develop solutions by following standard software engineering principles and implement by using suitable programming languages and platforms

Course Code	Course Title						Core/Elective
PC 655 AD	**Big Data Analytics Lab**						CORE
Prerequisite	Contact Hours Per Week				CIE	SEE	Credits
	L	T	D	p			
				2	30	70	1

Course Objectives

- To provide the knowledge to setup a Hadoop Cluster
- To impart knowledge to develop programs using MapReduce Technique
- To learn file handling in HDFS
- To introduce Pig, PigLatin and HiveQL to process big data
- To learn machine learning operations using Mahout Hadoop
- To introduce NoSQL databases

Course Outcomes

After completing this course, the student will be able to:

1. Understand Hadoop working environment
2. Work with big data applications in multi node clusters
3. Write scripts using Pig to solve real world problems
4. Write queries using Hive to analyse the datasets
5. Model and build a recommendation system using Mahout Hadoop and Apply big data and echo system techniques for real world

List of Experiments to be performed

1. Understanding and using basic HDFS commands
2. Word count application using Mapper Reducer on single node cluster
3. Analysis of Weather Dataset on Multi node Cluster
4. Working with files in Hadoop file system: Reading, Writing and Copying
5. Writing User Defined Functions/Eval functions for filtering unwanted data in Pig
6. Retrieving user login credentials from /etc/passwd using Pig Latin
7. Working with HiveQL.
8. Writing User Defined Functions in Hive
9. Perform classification & clustering in Mahout Hadoop
10. Building a Mahout Recommendation System on a Hadoop Cluster

Suggested Books:

1. Tom White, "Hadoop: The Definitive Guide", 4th Edition, O'Reilly Media Inc, April 2015. Alan Gates, "Programming Pig", O'Reilly MediaInc,2011.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Outcomes (CO's):

SUBJECT NAME: Big Data Analytics Lab **CODE: PC655AD**

SEMESTER: VI

CO No.	Course Outcomes	Taxonomy Level
PC655AD.1	Comprehend the operational framework of Hadoop	Understanding
PC655AD.2	Demonstrate proficiency in managing large-scale data tasks within multi-node clusters	Applying
PC655AD.3	Formulate scripts using Pig to address practical challenges	Applying
PC655AD.4	Create queries using Hive to scrutinize datasets	Analyzing
PC655AD.5	Construct and devise a recommendation system utilizing Mahout Hadoop, and employ advanced big data techniques for real-world scenarios	Creating

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the program / experiment details.
3. Student should enter into the laboratory with:
    a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.
    b. Laboratory Record updated up to the last session experiments.
    c. Formal dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out. If anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

Head of the Department **Principal**

CODE OF CONDUCT FOR THE LABORATORY

- All students must observe the dress code while in the laboratory
- Footwear is NOT allowed
- Foods, drinks and smoking are NOT allowed
- All bags must be left at the indicated place
- The lab timetable must be strictly followed
- Be PUNCTUAL for your laboratory session
- All programs must be completed within the given time
- Noise must be kept to a minimum
- Workspace must be kept clean and tidy at all time
- All students are liable for any damage to system due to their own negligence
- Students are strictly PROHIBITED from taking out any items from the laboratory
- Report immediately to the lab programmer if any damages to equipment

BEFORE LEAVING LAB:

- Arrange all the equipment and chairs properly.
- Turn off / shut down the systems before leaving.
- Please check the laboratory notice board regularly for updates.

Lab In – charge

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LIST OF EXPERIMENTS

5.	Writing User Defined Functions/Eval functions for filtering unwanted data in Pig			44	
6.	Retrieving user login credentials from /etc/passwd using Pig Latin			51	
7.	Sqoop installation and sqoop workouts			52	
8.	i) Hive installation ii) Working with HiveQL			55	
9.	HiveQL - Functions			58	
10.	Perform classification & clustering in Mahout Hadoop,Building a Mahout Recommendation System on a Hadoop Cluster			60	

ADDITIONAL EXPERIMENTS

SI.No.	Name of the Experiment	Date of Experiment	Date of Submission	Page No	Faculty Signature
9	Installing Hadoop			7	
10	HDFS Commands			10	
11	sqoop workout			55	

What is Big data?

Big data is a phrase or methodology that describes the aquire, cure, process and store the humongous volume of data that is limited to handled by the traditional systems with in the stipulated period of time.

Big data is simply the large sets of data that businesses and other parties put together to serve specific goals and operations. Big data can include many different kinds of data in many different kinds of formats. As a rule, it's raw and unsorted until it is put through various kinds of tools and handlers.

Big data usually includes data sets with sizes beyond the ability of commonly used softwaretools to capture, curate, manage, and process data within a tolerable elapsed time, extremely handleslarge data sets that may be analysed computationally to reveal patterns, trends, and associations,especially relating to human behaviour and interactions.

Why Bigdata draws market attention:

Every industry has its own particular big data challenges. Banks need to analyze streaming transactions in real time to quickly identify potential fraud. Utility companies need to analyse energy usage data to gain control over demand. Retailers need to understand the social sentiment around their products and markets to develop more effective campaigns and promotions.

Why Bigdata draws market attention:

Every industry has its own particular big data challenges. Banks need to analyze streaming transactions in real time to quickly identify potential fraud. Utility companies need to analyse energy usage data to gain control over demand. Retailers need to understand the social sentiment around their products and markets to develop more effective campaigns and promotions.

Size hierarchy

GB Gigabyte

TB

Terabyte

(1024

GB)PB

Petabyte

(1024 TB)

EB

Exabyte

(1024 PB)

ZB

Zetabyte

(1024 EB)

YB

Yotabyte

(1024 ZB)

Big Data

Statistics

Common

Data Sets

- 3.4 Zetabytes of data exist in the digital universe today.
- Data is doubling every 2 years, 90% of world's data volume is created in past 2 years.

Characteristics:

- Volume – huge volume of machine and man made data such as logs, click events, media filesand historical warehouse data.

  Eg . FB, Historical data (Bank example, Airlines example) etc.

- Velocity – Speed at which data is generated, captured, processed and stored. Eg. Tweets, click stream events, GPS Geospatial data etc.

- Variety – not only structured, it can accommodate un structured, semi structured, media, logs etc.
  Eg. STB logs, logs generated from different devices.
- Veracity – Trustworthy of the data we receive, how much true data we receive. Data accuracy is based on the veracity of source data that we receive at the very lower granular level. **Eg.**Website click steam logs.

Big data tools (other than hadoop):

Spark – In memory computing provided with Low latency SQL solution standalone and Hadoop, providesmachine learning libraries and graph database solution in a single place.

MongoDB – Document oriented DB, stores JSON (Java script object notation data).

TD Aster – Teradata uses mapreduce for the MPP store and process.

Cloud MapReduce - Initially developed at Accenture Technology Labs, Cloud Map reduce can be broadly defined as Mapreduce Implementation on Amazon Cloud OS. When compared with other open source implementation, it is completely different architecture than others.

PROGRAM 1

(i) Perform setting up and Installing Hadoop
in its three operatingmodes:

- Standalone
- Pseudo distributed
- Fully distributed

(ii) Use web based tools to monitor your Hadoop setup.
(iii) Working in the local environment commands

There are three modes in which Hadoop can be installed and run. They are:-

1. Standalone Mode
2. Pseudo Distributed Mode
3. Fully Distributed Mode

Standalone Mode

In this mode, there are no Hadoop Daemons (NameNode, DataNode, Secondary NameNode, JobTracker &TaskTracker) that are running in the background.

As a result you will,

- Not have NameNode storing meta-data information.
- Not have a DataNode, as there will be no HDFS. The file will be stored locally on the hard disk.
- Not have a TaskTracker sending status reports the JobTracker.
- Not have a JobTracker as there are no TaskTrackers to manage.

As the name suggests, everything in standalone mode runs in a single JVM (single machine). It is best suitedwhen you want to test your program for bugs with small input (stored locally). It is also known as the **LocalJobRunner** mode.

Note – Only Standalone mode supports quick testing of incremental changes that you make to the code.

Pseudo Distributed Mode

This mode helps you mimic a multi-server installation on a single machine. Pseudo Distributed mode also runson a single machine, but it has all the daemons running in a separate process. Also it will have the files stored on HDFS and not on the local machine. Thus in be seen as small scale implementation before you run on an actual cluster with 1000s of nodes.

Fully Distributed Mode

As the name suggests, this mode involves the code running on an actual

Hadoop cluster. It is mode in whichyou see the actual power of Hadoop, when you run your code against a very large input on 1000s of servers.

It is always difficult to debug a MapReduce program as you have Mappers running on different machine withdifferent piece of input. You can never know where the Mappers are going to run eventually. Also with large inputs, it is likely that the data will be irregular in its format.

Development Tip – So before you run your code on a real Hadoop cluster, following a few steps can make yourlife easy with MapReduce.

1. Always Unit Test your code first.
2. Start with a very small portion of the input.
3. Test locally (in LocalJobRunner mode) with the small input.
4. Test in Pseudo Distributed mode with daemons running (to view the performance of your code with the small input).
5. Finally if you are satisfied with running you code and its performance, you are ready to run it in FullyDistributed Mode on a real cluster.

Hadoop software can be installed in three modes of operation:

- **Stand Alone Mode**: Hadoop is a distributed software and is designed to run on a commodity of machines. However, we can install it on a single node in stand-alone mode. In this mode, Hadoop software runs as a single monolithic java process. This mode is extremely useful for debugging purpose. You can first test run your Map-Reduce application in this mode on small data, before actually executing it on cluster with big data.
- **Pseudo Distributed Mode**: In this mode also, Hadoop software is installed on a Single Node. Various daemons of Hadoop will run on the same machine as separate java processes. Hence all the daemons namely NameNode, DataNode, SecondaryNameNode, JobTracker, TaskTracker run on single machine.
- **Fully Distributed Mode**: In Fully Distributed Mode, the daemons NameNode, JobTracker, SecondaryNameNode (Optional and can be run on a separate node) run on the Master Node. The daemons DataNode and TaskTracker run on the Slave Node.

Hadoop Installation: Ubuntu Operating System in stand-alone mode

Steps for Installation

1. sudo apt-get update

2. In this step, we will install latest version of JDK (1.8) on the machine.

The Oracle JDK is the official JDK; however, it is no longer provided by Oracle as a defaultinstallation for Ubuntu. You can still install it using apt-get.

To install any version, first execute the following commands:

a. sudo apt-get install python-software-properties

b. sudo add-apt-repository ppa:webupd8team/java

c. sudo apt-get update

Then, depending on the version you want to install, execute one of the following commands:Oracle JDK 7: sudo apt-get install oracle-java7-installer

Oracle JDK 8: sudo apt-get install oracle-java8-installer

3. Now, let us setup a new user account for Hadoop installation. This step is optional, butrecommended because it gives you flexibility to have a separate account for Hadoop

installation by separating this installation from other software installation

a. sudo adduser hadoop_dev (Upon executing this command, you will prompted to enter thenew password for this user. Please enter the password and enter other details. Don't forgetto save the details at the end)

b. su - hadoop_dev (Switches the user from current user to the new user created i.eHadoop_dev)

4. Download the latest Hadoop distribution.

a. Visit this URL and choose one of the mirror sites. You can copy the download link andalso use "wget" to download it from command prompt:

Wget http://apache.mirrors.lucidnetworks.net/hadoop/common/hadoop-2.7.0/hadoop-2.7.0.tar.gz

5. Untar the file :

tar xvzf hadoop-2.7.0.tar.gz

6. Rename the folder to hadoop2

mv hadoop-2.7.0 hadoop2

7. Edit configuration file /home/hadoop_dev/hadoop2/etc/hadoop/hadoop-env.sh and setJAVA_HOME in that file.

a. vim /home/hadoop_dev/hadoop2/etc/hadoop/hadoop-env.sh

b. uncomment JAVA_HOME and update it following line:

export JAVA_HOME=/usr/lib/jvm/java-8-oracle (Please check for your relevant java installation and set this value accordingly. Latest versions of Hadoop require > JDK1.7)

8. Let us verify if the installation is successful or not(change to home directory cd /home/hadoop_dev/hadoop2/):

a. bin/hadoop (running this command should prompt you with various options)

9. This finishes the Hadoop setup in stand-alone mode.

10. Let us run a sample hadoop programs that is provided to you in the download package:

$ mkdir input (create the input directory)

$ cp etc/hadoop /*.xml input (copy over all the xml files to input folder)

$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.0.jar grep input output 'dfs[a-z.]+' (grep/find all the files matching the pattern ‘dfs[a-z.]+’ and copy those filesto output directory)

$ cat output/* (look for the output in the output directory that Hadoop creates for you).

Hadoop Installation: Psuedo Distributed Mode(Locally)

Steps for Installation

1. Edit the file /home/Hadoop_dev/hadoop2/etc/hadoop/core-site.xml as below:

```
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost:9000</value>
</property>
</configuration>
```

Note: This change sets the namenode ip and port.

2. Edit the file /home/Hadoop_dev/hadoop2/etc/hadoop/hdfs-site.xml as below:

```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
</configuration>
```

Note: This change sets the default replication count for blocks used by HDFS.

3. We need to setup password less login so that the master will be able to do a password-less ssh to start the daemons on all the slaves.

Check if ssh server is running on your host or not:

a. ssh localhost (enter your password and if you are able to login then ssh server is running)

b. In step a. if you are unable to login, then install ssh as follows:

```
sudo apt-get install ssh
```

c. Setup password less login as below:

i. `ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa`

ii. `cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys`

4. We can run Hadoop jobs locally or on YARN in this mode. In this Post, we will focus on

running the jobs **locally.**

5. Format the file system. When we format namenode it formats the meta-data related todata-nodes. By doing that, all the information on the datanodes are lost and they can bereused for new data:

a. bin/hdfs namenode –format

6. Start the daemons

a. sbin/start-dfs.sh (Starts NameNode and DataNode)

You can check If NameNode has started successfully or not by using the following webinterface: http://0.0.0.0:50070 . If you are unable to see this, try to check the logs in the /home/hadoop_dev/hadoop2/logs folder.

7. You can check whether the daemons are running or not by issuing Jps command.

8. This finishes the installation of Hadoop in pseudo distributed mode.

9. Let us run the same example we can in the previous blog post:

i) Create a new directory on the hdfs

```
bin/hdfs dfs -mkdir –p /user/hadoop_dev
```

ii) Copy the input files for the program to hdfs:

```
bin/hdfs dfs -put etc/hadoop input
```

iii) Run the program:

```
bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar grep input output
'dfs[a-z.]+'
```

iv) View the output on hdfs:

```
bin/hdfs dfs -cat output/*
```

10. Stop the daemons when you are done executing the jobs, with the below command:

```
sbin/stop-dfs.sh
```

Hadoop Installation – Psuedo Distributed Mode(YARN)

Steps for Installation

1. Edit the file /home/hadoop_dev/hadoop2/etc/hadoop/mapred-site.xml as below:

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

2. Edit the fie /home/hadoop_dev/hadoop2/etc/hadoop/yarn-site.xml as below:

```
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
</configuration>
```

Note: This particular configuration tells MapReduce how to do its shuffle. In this case ituses the mapreduce_shuffle.

3. Format the NameNode:

```
bin/hdfs namenode –format
```

4. Start the daemons using the command:

```
sbin/start-yarn.sh
```

This starts the daemons ResourceManager and NodeManager.

Once this command is run, you can check if ResourceManager is running or not by visitingthe following URL on browser : http://0.0.0.0:8088 . If you are unable to see this, check for the logs in the directory: /home/hadoop_dev/hadoop2/logs

5. To check whether the services are running, issue a jps command. The following shows allthe services necessary to run YARN on a single server:

$

j

p

s

1

5

9

3

3

J

p

s

15567 ResourceManager

15785 NodeManager

6. Let us run the same example as we ran before:

i) Create a new directory on the hdfs

```
bin/hdfs dfs -mkdir –p /user/hadoop_dev
```

ii) Copy the input files for the program to hdfs:

bin/hdfs dfs -put etc/hadoop input

iii) Run the program:

bin/yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar grep input output

'dfs[a-z.]+'

iv) View the output on hdfs:

bin/hdfs dfs -cat output/*

7. Stop the daemons when you are done executing the jobs, with the below command:

sbin/stop-yarn.sh

Working in the local environment:

1.Creating Directory:mkdir [directory name]

$mkdir BDA

2.Moving into Directory

$cd BDA

3.Displaying the Directory

Shortlist: ls
Longlist: ls –l
Longlist all files and Directory: ls –lart
Listing hidden files: ls –a

4.creating a files

$vi cse1:hi how r u(save and quit)
$echo “welcome to my class” >> cse2
$touch cse3(create empty file)

5.display content of a file:cat [filename]

$cat cse1
Output:hi how r u
$cat cse2
Output:welcome to my class

6.File operation

Copy: cp [source filename] [destination filename]

$echo ”AAAA” >> CS
$echo ”BBBB” >> DS

$cp CS DS

$cat CS
Output:AAAA
$cat DS
Outpu:AAAA

Move:mv [source filename] [destination filename]

$echo ”CCCC” >> CS1
$echo ”DDDD” >> DS2

$mv CS1 DS2
$cat CS1
Output: no file
$cat DS2
Output: CCCC

Delete:

Remove files: rm [filename]
$rm DS2

Remove directory: rm –r [directory name]
$rm –r BDA

7.Disk size commands

1.Disk free
df -k path
2.Disk usage
du -k path

8.Environment variables
env

9.History of the commands used
history

10.Grep
grep sometext filename.txt

11.Execute a shell script
vi scriptname.bsh
a='hi how are you'
echo $a
bash scriptname.bsh

12.Variables
a='hi how are you'
echo $a
export $a

13.Compression
gzip filename
gunzip filename.gz

14.Directory commands
You can go in your home directory anytime using the following command −
cd ~

To go in your last directory you can use following command −
cd -

To go to the parent directory
cd ..
cd ../..

Absolute path
Fully qualified path start with '/' provided from root till the child.
cd /
mkdir /home/hduser/dirname

Week 2:

a. Understanding and using basic HDFS commands

b. Working with files in Hadoop file system: Reading, Writing and Copying

HDFS File System Commands:

Apache Hadoop has come up with a simple and yet basic Command Line interface, a simple interface to access the underlying Hadoop Distributed File System. In this section we will introduce you to the basic and the most useful HDFS File System Commands which will be more or the like similar to UNIX file system commands. Once the Hadoop daemons are started and UP and Running, HDFS file system is ready to be used. The file system operations like creating directories, moving files, adding files, deleting files, reading files and listing directories can be done seamlessly on the same.

We can get list of FS Shell commands with below command.

$

hadoo

p fs -

help

Usage: hadoop fs [generic options]

[-appendToFile <localsrc> ... <dst>]

[-cat [-ignoreCrc] <src> ...]

[-checksum <src> ...]

[-chgrp [-R] GROUP PATH...]

[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]

[-chown [-R] [OWNER][:[GROUP]] PATH...]

[-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]

[-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]

[-count [-q] [-h] <path> ...]

[-cp [-f] [-p | -p[topax]] <src> ... <dst>]

[-createSnapshot <snapshotDir> [<snapshotName>]]

[-deleteSnapshot <snapshotDir> <snapshotName>]

[-df [-h] [<path> ...]]

[-du [-s] [-h] <path> ...]

[-expunge]
[
-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]

[-getfacl [-R] <path>]

[-getfattr [-R] {-n name | -d} [-e en] <path>]

[-getmerge [-nl] <src> <localdst>]

[-help [cmd ...]]

[-ls [-d] [-h] [-R] [<path> ...]]

[-mkdir [-p] <path> ...]

[-moveFromLocal <localsrc> ... <dst>]

[-moveToLocal <src> <localdst>]

[-mv <src> ... <dst>]

[-put [-f] [-p] [-l] <localsrc> ... <dst>]

[-renameSnapshot <snapshotDir> <oldName> <newName>]

[-rm [-f] [-r|-R] [-skipTrash] <src> ...]

[-rmdir [--ignore-fail-on-non-empty] <dir> ...]

[-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>]|[--set <acl_spec> <path>]]

[-setfattr {-n name [-v value] | -x name} <path>]

[-setrep [-R] [-w] <rep> <path> ...]

[-stat [format] <path> ...]

[-tail [-f] <file>]

[-test -[defsz] <path>]

[-text [-ignoreCrc] <src> ...]

[-touchz <path> ...]

[-usage [cmd ...

1. **mkdir:**

This is no different from the UNIX **mkdir** command and is used to create a directory on a HDFS environment. The option –**p** is used to mention not to fail if the directory already exists.

Syntax$ hadoop fs -mkdir /user/hadoop/

$hadoop fs -mkdir[p] Usage:
$ hadoop fs -mkdir /user/hadoop/

$ hadoop fs -mkdir /user/data/

In order to create subdirectories, the parent directory needs to be existing already. If the condition is not met then 'No such file or directory' message will be returned.

2. **ls:**

This is no different from the UNIX **ls** command and it is used for listing the directories present under a specific directory in an HDFS system. The –**lsr** command may be used for the recursive listing of the directories and files under a specific folder.

The –**d** option is used to list the directories as plain files

The –**h** option is used to format the sizes of files into a human readable manner than just number of bytes

The –**R** option is used to recursively list

the contents of directories**Syntax:**

$ hadoop fs -ls [-d] [-h] [-R]

Usage:

$ hadoop fs -ls /

$ hadoop fs -lsr /

The command above will match the specified file pattern, directory entries are of the form (as shown below)

permissions - userId groupId sizeOf Directory(in bytes)
modification Date(yyyy-MM-dd HH:mm) directoryName

3. **put:**

This command is used to copy files from the local file system to the HDFS filesystem. This is very muchsimilar to the –copy FromLocal command. Copying of the files fails if the file already exists, unless the
–**f** flag is given to the command. This overwrites the destination if the file already exists before the copyThe –**p** flag preserves the access, modification time, ownership and the mode

Syntax:

$ hadoop fs -put [-f]

[-p] Usage:

$ hadoop fs -put sample.txt /user/data/

4. **get:**

This command Copies files from HDFS file system to the local file system, just the opposite of thecommand that we have seen just now. This is similar to the –copyToLocal command

Usage:

$ hadoop fs -get /user/data/sample.txt workspace/

5. **cat:**

This command is similar to the UNIX **cat** command and is used for displaying the contents of a file on theconsole.

Usage:

$ hadoop fs -cat /user/data/sampletext.txt

6. **cp:**

This command is similar to the UNIX **cp** command, and it is used for copying files from one directory toanother directory within the HDFS file system.

Usage:

$ hadoop fs -cp /user/data/sample1.txt /user/hadoop1

$ hadoop fs -cp /user/data/sample2.txt /user/test/in1

7. **mv:**

This command is similar to the UNIX **mv** command, and it is used for moving a file from one directory toanother directory within the HDFS file system.

Usage:

$ hadoop fs -mv /user/hadoop/sample1.txt /user/text/

8. **rm:**

This command is similar to the UNIX **rm** command, and it is used for removing a file from the HDFS filesystem. The command –rmr can be used to delete files recursively.

Directories can't be deleted by the –**rm** command, we need to use the –**rm r** (recursive remove) command to delete directories and files inside them. Only files can be removed using the –**rm** command.

The –**skipTrash** option is used to bypass the trash, if enabled, and then it immediately deletes the sourceas mentioned in the tag.

The –**f** option is used to mention that if there is no file existing, then not to display any diagnosticmessage or even to modify the exit status to reflect on showing up an error.

The –**rR** option is used to recursively delete directories

Syntax:

$ hadoop fs -rm [-f] [-r|-R]

[skipTrash]

Usage:

$ hadoop fs -rm -r /user/test/sample.txt

9. **getmerge:**

This is the most important and the most useful command on the HDFS filesystem, when trying to read the contents of a MapReduce job or PIG job's output files. This is used for merging a list of files in a directory on the HDFS filesystem into a single local file on the local filesystem.

Usage:

$ hadoop fs -getmerge /user/data

10. **touchz:**

This command can be used to create a file of zero bytes size in HDFS filesystem.

Usage:

$ hadoop fs -touchz URI

11. **test:**

This command is used to test a HDFS file's existence of zero length of the file or whether if it is a directory or not.

The –**d** option is used to check whether if it is a directory or not, returns 0 if it is a

directoryThe –**e** option is used to check whether the exists or not, returns 0 if the exists

The –**f** option is used to check whether the is a file or not, returns 0 if the file exists

The –**s** option is used to check whether the file size is greater than 0 bytes or not, returns 0 if the size isgreater than 0 bytes

The –**z** option is used to check whether the file size is zero bytes or not. If the file size is zero bytes, thenreturns 0 or else returns 1.

Usage:

$ hadoop fs -test -[defsz] /user/test/test.txt

12. appendToFile:

This command appends the contents of all the given local files to the provided destination file on the HDFS filesystem. The destination file will be created if it is not existing earlier. If the is -, then the input isread from **stdin**.

Syntax:

$ hadoop fs

appendToFile

Usage:

$ hadoop fs -appendToFile derby.log data.tsv /in/appendfile

$ hadoop fs -cat /in/appendfile

13. tail:

This command is used to show the last 1KB of the file.

The –**f** option is used to the show appended data as the file grows

Syntax:

$ hadoop fs -tail

[f]

Usage:

user@tri03ws-386:~$ hadoop fs -tail

/in/appendfileSun Oct 15 14:41:10 IST

2017:

14. **stat:**

This command is used to print the statistics about the file / directory at in the specified format. Format accepts file size in blocks (%b), group name of the owner (%g) and the file name (%n), block size (%o), replication (%r), user name of the owner (%u), modification date (%y, %Y)

Syntax:

$ hadoop fs -stat

[format]Usage:

user@tri03ws-386:~$ hadoop fs -stat

/in/appendfile2014-11-26 04:57:04

user@tri03ws-386:~$ hadoop fs -stat %Y

/in/appendfile1416977824841

user@tri03ws-386:~$ hadoop fs -stat %b

/in/appendfile20981

user@tri03ws-386:~$ hadoop fs -stat %r

/in/appendfile1

user@tri03ws-386:~$ hadoop fs -stat %o

/in/appendfile134217728

15. **setfattr:**

This command sets an extended attribute name and value for a file or

directory on the HDFS filesystem.The –**n** option is used to provide the

extended attribute name

The –**x** option is used to remove the extended attribute , file or directory

The –**v** option is used to provide the extended attribute value. There are 3 different encoding methods available for the value.

1. If the argument is enclosed in double quotes, then the value is the string inside the quotes
2. If the argument is prefixed with 0x or 0X, then it is taken as a hexadecimal number
3. If the argument begins with 0s or 0S, then it is

taken as a Base64 encoding**Syntax:**

$ hadoop fs -setfattr {-n name [-v value] | -x name}

18. **df:**

This command is used to show the capacity, free and the used space available on the HDFS filesystem. If the filesystem has multiple partitions and if there is no path is mentioned to any specific partition, then the status of the root partition will be displayed for us to know.

The –**h** option is used to format the sizes of the files in a human readable manner rather than number of bytes.

Syntax:

$ hadoop fs -df [-h] [...]

19. **du:**

This command is used to show the amount of space in Bytes that has been used by the files that match the specified file pattern. Even without the –s option, this only shows the size summaries one level deepin the directory.

The –**s** option is used to show the size of each individual file that matches the pattern, shows the total (summary) size

The –**h** option is used to format the sizes of the files in a human readable manner rather than number of bytes.

Syntax:

$ hadoop fs -du [-s] [-h]

20. **count**This command is used to count the number of directories, files and bytes under the path that match theprovided file pattern.

Syntax:

$ hadoop fs -count [-q]

The output columns are as follows:

1. DIR_COUNT FILE_COUNT CONTENT_SIZE FILE_NAME
2. QUOTA REMAINING_QUOTA SPACE_QUOTA REMAINING_SPACE_QUOTA
3. DIR_COUNT FILE_COUNT CONTENT_SIZE FILE_NAME

21.chgrp:

This command is used to change the group of a file or a path.

Syntax:

$ hadoop fs -chgrp [-R] groupname

Permissions
Types - Owner, group and others
ls -l /home/hduser

-rwxr-xr-- 1 inceptez inceptez 1024 Nov 2 00:10 myfile
drwxr-xr--- 1 inceptez inceptez 1024 Nov 2 00:10 mydir
Here first column represents different access mode ie. permission associated with a file or directory.
The permissions are broken into groups of threes, and each position in the group denotes a specific permission, in this order: read (r), write (w), execute (x) −

- The first three characters (2-4) represent the permissions for the file's owner. For example - rwxr-x**r--** represents that onwer has read (r), write (w) and execute (x) permission.
- The second group of three characters (5-7) consists of the permissions for the group to which the file belongs. For example -rwxr-x**r--** represents that group has read (r) and execute (x) permission but no write permission.
- The last group of three characters (8-10) represents the permissions for everyone else. For example -rwxr-x**r--** represents that other world has read (r) only permission.

Change mode:

Change permission of the owner and provide read and write access.
chmod u+rw filename
Change permission of the group and provide read and write access.
chmod g+rw filename
Change permission of the others and provide read and write access.
chmod o+rw filename

Number	**Octal Permission Representation**	**Ref**
0	No permission	---
1	Execute permission	--x
2	Write permission	-w-
4	Read permission	r--

Changing owners and Groups

Change owner from hduser to inceptez
sudo chown inceptez filename
Change group from hduser to inceptez
sudo chgrp inceptez filename

Common default directories:

Directory	Description
/	This is the root directory which should contain only the directories needed at the top level of the file structure.
/bin	This is where the executable files are located. They are available to all user.
/dev	These are device drivers.
/etc	Supervisor directory commands, configuration files, disk configuration files, valid user lists, groups, ethernet, hosts, where to send critical messages.
/lib	Contains shared library files and sometimes other kernel-related files.
/home	Contains the home directory for users and other accounts.
/usr	Used for miscellaneous purposes, or can be used by many users. Includes administrative commands, shared files, library files, and others
/var	Typically contains variable-length files such as log and print files and any other type of file that may contain a variable amount of data
/sbin	Contains binary (executable) files, usually for system administration. For example *sbin utility for useradd commands etc.*.

21. chmod:

This command is used to change the permissions of a file, this command works as similar to that ofLINUX's shell command **chmod** with a few exceptions.

The –**R** option is used to modify the files recursively and it is the only option that is being supportedcurrently.

The is the same as the mode used for the shell command. The letters that are recognized are '**rwxXt**'.

The is the mode specified in 3 or 4 digits. The first may be 0 or 1 to turn the sticky bit OFF or ONrespectively. Unlike the shell command, it is not at all possible

to specify only part of the mode.

Syntax:

$ hadoop fs -chmod [-R] PATH

22. **chown**

This command is used to change the owner and group of a file. This command is similar to theshell's **chown** command with a few exceptions.

If only the owner of the group is specified then only the owner of the group is modified via this command. The owner and group names may only consists of digits, alphabets and any of the characters mentioned here [-_./@a-zA-Z0-9]. The names thus specified are case sensitive as well.

It is better to avoid using '.' to separate user name and the group just the way LINUX allows it. If the usernames have dots in them and if you are using local file system, you might see surprising results since the shell command chown is used for the local file alone.

	Input	Set of data	Bus, Car, bus, car, train, car, bus, car, train, bus, TRAIN,BUS, buS,caR, CAR, car, BUS, TRAIN
	Output	Convert into another setof data (Key,Value)	(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1), (TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1),(car,1), (BUS,1), (TRAIN,1)

Reduce Function – Takes the output from Map as an input and combines those datatuples into a smaller set of tuples.

Example – (Reduce function in Word Count)

	Input **(output of Map function)**	Set of Tuples	(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1), (TRAIN,1),(BUS,1), (buS,1), (caR,1),(CAR,1), (car,1), (BUS,1), (TRAIN,1)	

Output	Converts into smaller set oftuples	(BUS,7), (CAR,7), (TRAIN,4)

Work Flow of Program:

Workflow of MapReduce consists of 5 steps

1. **Splitting** – The splitting parameter can be anything, e.g. splitting byspace, comma, semicolon, or even by a new line (‘\n’).
2. **Mapping** – as explained above
3. **Intermediate splitting** – the entire process in parallel on different clusters. In order to group them in “Reduce Phase” the similar KEYdata should be on same cluster.
4. **Reduce** – it is nothing but mostly group by phase
5. **Combining** – The last phase where all the data

(individual result set from each cluster) is combine together to form a Result

Now Let's See the Word Count Program in Java

Fortunately we don't have to write all of the above steps, we only need to write the splitting parameter, Map function logic, and Reduce function logic.The rest of the remaining steps will execute automatically.

Make sure that Hadoop is installed on your system with java idk

Steps

Step 1. Open Eclipse> File > New > Java Project >(Name it –MRProgramsDemo) > Finish

Step 2. Right Click > New > Package (Name it - PackageDemo) > FinishStep 3. Right Click on Package > New > Class (Name it - WordCount) Step 4. Add Following Reference Libraries – Right Click on Project > Build Path> Add External Archivals

- /usr/lib/hadoop-0.20/**hadoop-core.jar**
- Usr/lib/hadoop-0.20/lib/**Commons-cli-1.2.jar**

Step 5. Type following Program :

```
package PackageDemo;
import java.io.IOException;
import
org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import
org.apache.hadoop.mapreduce.Mapper;
import
org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class WordCount {
public static void main(String [] args) throws Exception
{
Configuration c=new Configuration();
String[] files=new
GenericOptionsParser(c,args).getRemainingArgs(); Path
input=new Path(files[0]);
Path output=new Path(files[1]);
Job j=new Job(c,"wordcount");
j.setJarByClass(WordCount.class
);
j.setMapperClass(MapForWordCount.class
);
j.setReducerClass(ReduceForWordCount.clas
s); j.setOutputKeyClass(Text.class);
```

```
j.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(j, input);
FileOutputFormat.setOutputPath(j, output);
System.exit(j.waitForCompletion(true)?0:1);
}
public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable>{
public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException
{
String line = value.toString();
String[] words=line.split(",");
for(String word: words )
{
    Text outputKey = new
 Text(word.toUpperCase().trim());IntWritable
 outputValue = new IntWritable(1);
 con.write(outputKey, outputValue);
}
}
}
public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text, IntWritable>
{
public void reduce(Text word, Iterable<IntWritable> values, Context con) throws IOException,
InterruptedException
{
int sum = 0;
  for(IntWritable value : values)
  {
  sum += value.get();
  }
  con.write(word, new IntWritable(sum));
}
}
}
```

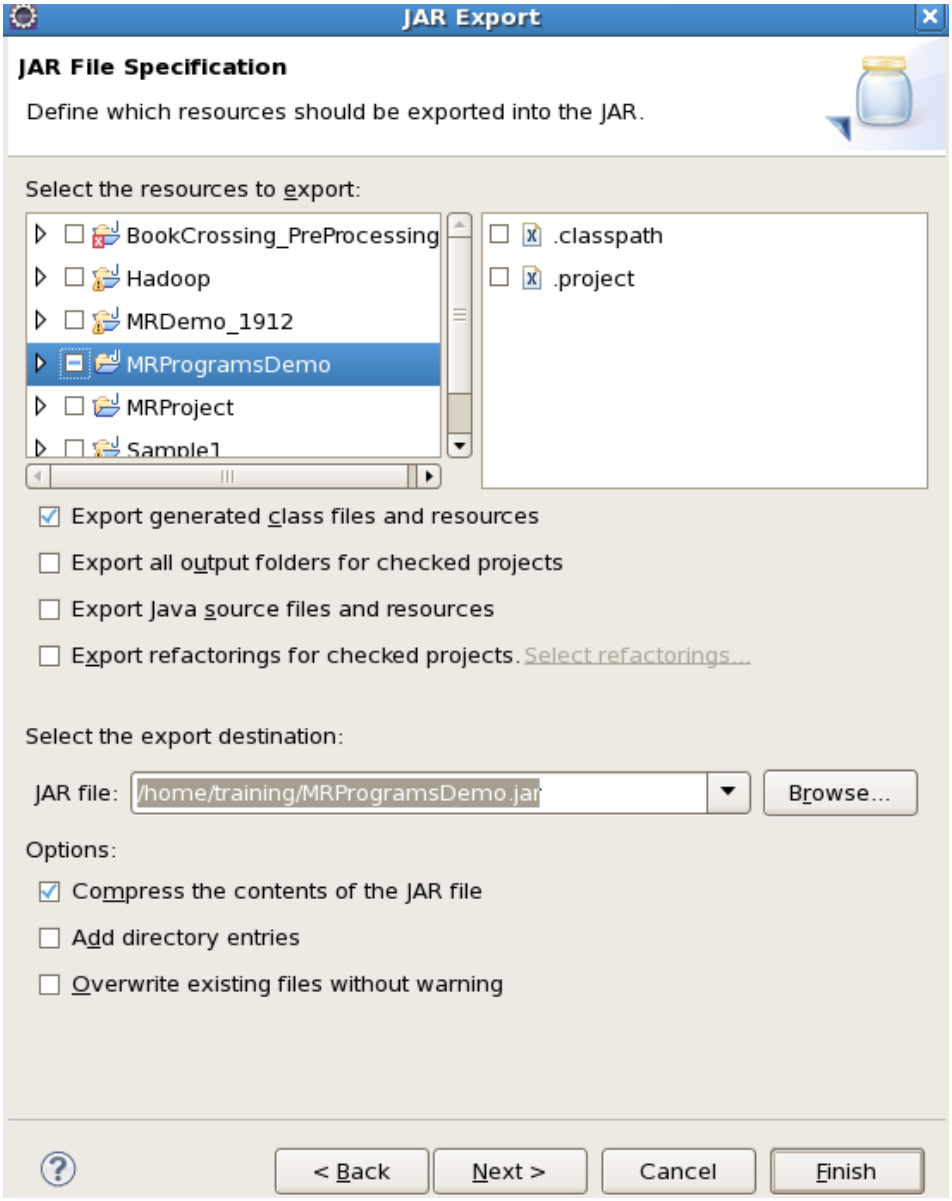
Explanation

The program consist of 3 classes:

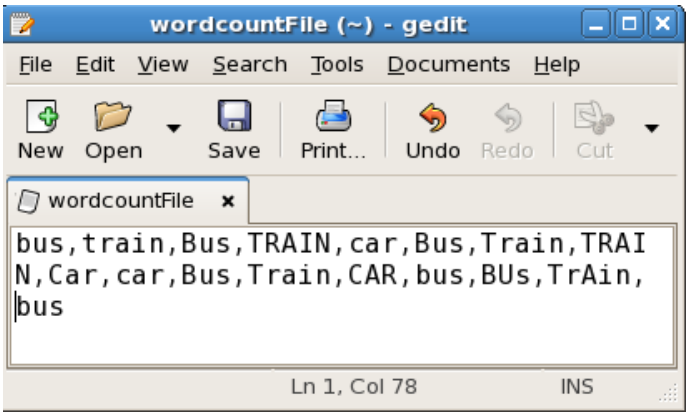
- Driver class (Public void static main- the entry point)
- Map class which **extends** public class Mapper<KEYIN,VALUEIN,KEYOUT,VALUEOUT> and implements theMap function.
- Reduce class which extends public class Reducer<KEYIN,VALUEIN,KEYOUT,VALUEOUT> and implements theReduce function.

Step 6. Make Jar File

Right Click on Project> Export> Select export destination as **Jar File** > next>Finish



Step 7: Take a text file and move it in HDFS



To Move this into Hadoop directly, open the terminal and enter the followingcommands:

```
[training@localhost ~]$ hadoop fs -put wordcountFile wordCountFile
```

Step 8 . Run Jar file

(hadoop jar jarfilename.jar packageName.ClassName PathToInputTextFilePathToOutputDirectry)

```
[training@localhost ~]$ hadoop jar MRProgramsDemo.jar PackageDemo.WordCount wordCountFile MRDir1
```

Step 9. Open Result

```
[training@localhost ~]$ hadoop fs -ls MRDir1Found
3 items
-rw-r--r--   1 training supergroup          0 2016-02-23 03:36 /user/training/MRDir1/_SUCCESS
drwxr-xr-x   - training supergroup          0 2016-02-23 03:36 /user/training/MRDir1/_logs
-rw-r--r--   1 training supergroup         20 2016-02-23 03:36 /user/training/MRDir1/part-r-00000
[training@localhost ~]$ hadoop fs -cat MRDir1/part-r-00000
BUS     7
CAR     4
TRAIN   6
```

Week 4: Analysis of Weather Dataset on Multi node Cluster

Weather sensors collecting data every hour at many locations across the globe gather a large volume of log data, which is a good candidate for analysis with MapReduce, since it is semi structured and record-oriented.

```
import java.io.IOException;

import java.util.Iterator;

import   org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable; import

org.apache.hadoop.io.Text;


import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import

org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import

org.apache.hadoop.mapreduce.lib.output.TextOutputFormat; import

org.apache.hadoop.mapreduce.lib.input.TextInputFormat; import

org.apache.hadoop.mapreduce.Job;

import   org.apache.hadoop.mapreduce.Mapper;
```

```
import org.apache.hadoop.mapreduce.Reducer;import
org.apache.hadoop.conf.Configuration;

public class MyMaxMin {


        //Mapper

        /**
        *MaxTemperatureMapper class is static and extends Mapper abstract classhaving
        four hadoop generics type LongWritable, Text, Text, Text.
        */


        public static class MaxTemperatureMapper extends
                        Mapper<LongWritable, Text, Text, Text> {

                /**
                * @method map
                * This method takes the input as text data type.
                * Now leaving the first five tokens,it takes 6th token is taken as temp_max and
                * 7th token is taken as temp_min. Now temp_max > 35 and temp_min < 10 arepassed
to the reducer.
                */

                @Override
                public void map(LongWritable arg0, Text Value, Context context)throws
```

```
                    IOException, InterruptedException

        {

        //Converting the record (single line) to String and storing it in a String variable
line


            String line = Value.toString();

          //Checking if the line is not empty

            if (!(line.length() == 0)) {

                 //date

                String date = line.substring(6, 14);

                //maximum temperature

                float temp_Max = Float
                                    .parseFloat(line.substring(39, 45).trim());

                //minimum temperature

                float temp_Min = Float
                                    .parseFloat(line.substring(47, 53).trim());

            //if maximum temperature is greater than 35 , its a hot day

                if (temp_Max > 35.0) {
                    // Hot day
```

```
                        context.write(new Text("Hot Day " + date),
                                new Text(String.valueOf(temp_Max)));
                    }

                    //if minimum temperature is less than 10 , its a cold day

                    if (temp_Min < 10) {
                        // Cold day
                        context.write(new Text("Cold Day " + date),
                                new Text(String.valueOf(temp_Min)));
                    }
                }
            }

        }

//Reducer

        /**
        *MaxTemperatureReducer class is static and extends Reducer abstract classhaving
        four hadoop generics type Text, Text, Text, Text.
        */

        public static class MaxTemperatureReducer extends
                Reducer<Text, Text, Text, Text> {
```

```
        /**
        * @method reduce
        * This method takes the input as key and list of values pair from mapper, itdoes
aggregation
        * based on keys and produces the final context.
        */
        public void reduce(Text Key, Iterator<Text> Values, Context context)throws
                        IOException, InterruptedException {

                //putting all the values in temperature variable of type String

                String temperature = Values.next().toString();
                context.write(Key, new Text(temperature));
        }
}

/**
* @method main
* This method is used for setting all the configuration properties.
* It acts as a driver for map reduce code.
*/

public static void main(String[] args) throws Exception {
```

```
//reads the default configuration of cluster from the configuration xml filesConfiguration conf = new Configuration();

        //Initializing the job with the default configuration of the clusterJob job =
        new Job(conf, "weather example");

        //Assigning the driver class name
        job.setJarByClass(MyMaxMin.class);

        //Key type coming out of mapper
        job.setMapOutputKeyClass(Text.class);

        //value type coming out of mapper
        job.setMapOutputValueClass(Text.class);

        //Defining the mapper class name job.setMapperClass(MaxTemperatureMapper.class);

        //Defining the reducer class name job.setReducerClass(MaxTemperatureReducer.class);

        //Defining input Format class which is responsible to parse the dataset into akey value
pair
        job.setInputFormatClass(TextInputFormat.class);

        //Defining output Format class which is responsible to parse the dataset into akey value
pair
        job.setOutputFormatClass(TextOutputFormat.class);
```

```
                //setting the second argument as a path
                in a path variablePath OutputPath =
                new Path(args[1]);

                //Configuring the input path from the filesystem into the job
                FileInputFormat.addInputPath(job, new Path(args[0]));

                //Configuring the output path from the filesystem into the job
                FileOutputFormat.setOutputPath(job, new Path(args[1]));

                //deleting the context path automatically from hdfs so that we
don't havedelete it explicitly
                OutputPath.getFileSystem(conf).delete(OutputPath);

                //exiting the job only if the flag
                value becomes false
                System.exit(job.waitForCompl
                etion(true) ? 0 : 1);

        }
}
```

Step 2

Import the project in eclipse IDE in the same way it was told in earlier guide and change the jarpaths with the jar files present in the lib directory of this

project.

Step 3

When the project is not having any error, we will export it as a jar file, same as we did in wordcount mapreduce guide. Right Click on the Project file and click on Export. Select jar file.Give the path where you want to save the file.Select the mail file by clicking on browse.
Click on Finish to export.

Step 4

You can download the jar file

directly using below link

temperature.jar

https://drive.google.com/file/d/0B2SFMPvhXPQ5RUlZZDZSR3FYVDA/view?usp=sharing

Download Dataset used

by me using below link

weather_data.txt

https://drive.google.com/file/d/0B2SFMPvhXPQ5aFVILXAxbFh6ejA/view?usp=sharing

Step 5

Before running the mapreduce program to check what it does, see that your cluster is up and allthe hadoop daemons are running.

Command: jps

Step 6

Send the weather dataset on to HDFS.

Command: hdfs dfs -put

Downloads/weather_data.txt /

Command: hdfs dfs -ls /

Step 7

Run the jar file.

Command: hadoop jar temperature.jar /weather_data.txt /output_hotandcold

Step 8

Check output_hotandcold directory in HDFS.

Now inside part-r-00000 you will get your output.

Week5: Writing User Defined Functions/Eval functions for filtering unwanted data in Pig

What is Apache Pig

Apache Pig is a high-level data flow platform for executing MapReduce programs of Hadoop. The language used for Pig is Pig Latin.

The Pig scripts get internally converted to Map Reduce jobs and get executed on data stored in HDFS. Apart from that, Pig can also execute its job in Apache Tez or Apache Spark.

Pig can handle any type of data, i.e., structured, semi-structured or unstructured and stores the corresponding results into Hadoop Data File System. Every task which can be achieved using

PIG can also be achieved using java used in MapReduce.

Pig Installation

Pig Installation with local metastore configuration

1) copy the hive tar file to your home path (/home/hduser/install) and extract using below command
   cd /home/hduser/install/
2) tar xvzf apache-pig-0.14.0-bin.tar.gz
3) sudo mv apache-pig-0.14.0-bin /usr/local/pig
4) Make an entry in the bash profile like below,
   vi ~/.bashrc

   export PIG_HOME=/usr/local/pig
   export PATH=$PATH:$PIG_HOME/bin

5) Then source the bash profile like below,
   source ~/.bashrc

6) $pig
   Open grant shell

Pig Latin commands:

Pig Latin statements are the basic constructs you use to process data using Pig. A Pig Latin statement is an operator that takes a relation as input and produces another relation as output. (This definition applies to all Pig Latin operators except LOAD and STORE which read data from and write data to the file system.) Pig Latin statements may include expressions and schemas. Pig Latin statements can span multiple lines and must end with a semi-colon (;). By default, Pig Latin statements are processed using multi-query execution. A Pig Latin program consists of a collection of statements.

LOAD :A LOAD statement to read data from the file system. This operator loads data from the file or directory. If a directory name is specified, it loads all the files in the directory into the relation. If Pig is run in the local mode, it searches for the directories on the local File System; while in the MapReduce mode, it searches for the files on HDFS.

DUMP :A DUMP statement to view results or a STORE statement to save the results. The DUMP operator is almost similar to the STORE operator, but it is used specially to display results on the command prompt rather than storing it in a File System like the STORE operator. DUMP behaves in exactly the same way as STORE, where the Pig Latin statements actually begin execution after encountering the DUMP operator.

PIG SCRIPTING:

$gedit text editor

Input

111,John,sales,Austin
222,Alex,Marketing,New york
333,Philp,Operation,Sacramento
444,Terry,sales,New york
555,Jessi,Development,Boston

Save the file

$hadoop fs -put Desktop/input /pigInput

$hadoop fs -cat /pigInput
Output:
111,John,sales,Austin
222,Alex,Marketing,New york
333,Philp,Operation,Sacramento
444,Terry,sales,New york
555,Jessi,Development,Boston

$pig
grunt>

Loading: Loads data from the file system.
grunt>employee = LOAD '/pigInput' using PigStorage(',') AS (ssn:chararray,name
:chararray,department:chararray,city:chararray);
grunt>dump employee;
output:
(111,John,sales,Austin)
(222,Alex,Marketing,New york)
(333,Philp,Operation,Sacramento)
(444,Terry,sales,New york)
(555,Jessi,Development,Boston)

Foreach: Generates transformation of data for each row as specified
grunt>emp_foreach = foreach employee generate name,department;

grunt>dump emp_foreach;
output:
(John,sales)
(Alex,Marketing)
(Philp,Operation)
(Terry,sales)
(Jessi,Development)

Filter: Selects tuples from a relation based on some condition.
grunt>emp_filter = filter employee by city == 'Austin';
grunt>dump emp_filter;
output:
(111,John,sales,Austin)

Order: Sorts a relation based on one or more fields.
grunt>emp_order = order employee by ssn desc;
grunt>dump emp_order;
output:
(555,Jessi,Development,Boston)
(444,Terry,sales,New york)
(333,Philp,Operation,Sacramento)
(222,Alex,Marketing,New york)
(111,John,sales,Austin)

Example2:
$gedit text editor
student
1,A
2,B
3,D
4,D

save the file

$hadoop fs -put Desktop/student /pigStudent
$hadoop fs -cat /pigStudent
output:
1,A
2,B
3,D
4,D

==>if we use pig command we get grant shell
open new terminal
typing pig command

$pig
grunt>

Loading: : Loads data from the file system.
grunt> s = load '/pigStudent' using PigStorage(',') AS (id:int,name:chararray);
grunt>dump s;
output:
1,A
2,B
3,D
4,D

Filter: Selects tuples from a relation based on some condition.
grunt> f =filter s by id>1;
grunt> dump f;
output:
(2,B)
(3,D)
(4,D)

sorting: Sorts a relation based on one or more fields.
grunt> rsort = order s by id desc;
grunt> dump rsort;
output:
(4,D)
(3,D)
(2,B)
(1,A)

grunt> sort = order s by id asc;
grunt> dump sort;
output:
1,A
2,B
3,D
4,D

Distint: Removes duplicate tuples in a relation.
grunt> distinct_data = DISTINCT s;

grunt> dump distinct_data;
output:
(1,A)
(2,B)
(3,D)
(4,D)

Foreach:
grunt> foreach_data = FOREACH s generate name,id;
grunt> dump foreach_data;
output:
(A,1)
(B,2)
(D,3)
(D,4)

AGGREGATE FUNCTIONS:

$gedit text editor
emp
101,a,b,1000,40
102,c,d,1500,25
103,e,f,2000,24
104,g,h,1500,24
105,i,j,3000,25

Save the file on desktop

$hadoop fs -put Desktop/emp /pigemp
$hadoop fs -cat /pigemp
output:
101,a,b,1000,40
102,c,d,1500,25
103,e,f,2000,24
104,g,h,1500,24
105,i,j,3000,25

Load: : Loads data from the file system.
grunt> a = load '/pigemp' using PigStorage(',') as (f1:int,f2:chararray,f3:chararray,f4:int,f5:int);
grunt> dump a;
output:
(101,a,b,1000,40)
(102,c,d,1500,25)
(103,e,f,2000,24)
(104,g,h,1500,24)

(105,i,j,3000,25)

Group: Groups the data in one or more relations.

```
grunt> emp_grp =GROUP a all;
grunt> dump emp_grp;
```

otput:

(all,{(,,,,),(105,i,j,3000,25),(104,g,h,1500,24),(103,e,f,2000,24),(102,c,d,1500,25),(101,a,b,1000,40)})

Count: Computes the number of elements in a bag, it ignores null.

```
grunt> emp_count = foreach emp_grp generate a;
grunt> dump emp_count;
```

output:

({(,,,,),(105,i,j,3000,25),(104,g,h,1500,24),(103,e,f,2000,24),(102,c,d,1500,25),(101,a,b,1000,40)})

```
grunt> emp_count = foreach emp_grp generate a.f1;
grunt> dump emp_count;
```

output:

({(),(105),(104),(103),(102),(101)})

Average: Computes the average of the numeric values in a single-column bag.

```
grunt> emp_avg = foreach emp_grp generate (a.f2,a.f4),AVG(a.f4);
grunt> dump emp_avg;
```

output:

(({(),(i),(g),(e),(c),(a)},{(),(3000),(1500),(2000),(1500),(1000)}),1800.0)

Maximum: Computes the maximum of the numeric values or chararrays in a single-column bag

```
grunt> emp_max = foreach emp_grp generate (a.f2,a.f4),MAX(a.f4);
grunt> dump emp_max;
```

output:

(({(),(i),(g),(e),(c),(a)},{(),(3000),(1500),(2000),(1500),(1000)}),3000)

Minimum: Computes the minimum of the numeric values or chararrays in a single-column bag.

```
grunt> emp_min = foreach emp_grp generate(a.f2,a.f4),MIN(a.f4);
grunt> dump emp_min;
```

output:

(({(),(i),(g),(e),(c),(a)},{(),(3000),(1500),(2000),(1500),(1000)}),1000)

LOWER: Converts all characters in a string to lower case.

```
grunt> emp_lwr =foreach a generate LOWER(f2);
```

grunt> dump emp_lwr;
output:
(a)
(c)
(e)
(g)
(i)

grunt> emp_lwr =foreach a generate(f1,f2), LOWER(f2);
grunt> dump emp_lwr;
OUTPUT:
((101,a),a)
((102,c),c)
((103,e),e)
((104,g),g)
((105,i),i)

UPPER: Returns a string converted to upper case.
grunt> emp_upr =foreach a generate(f1,f2), UPPER(f2);
grunt> dump emp_upr;
output:
((101,a),A)
((102,c),C)
((103,e),E)
((104,g),G)
((105,i),I)

GROUP: Groups the data in one or more relations.
grunt> b = group a by f1;
grunt> dump b;
output:
(101,{(101,a,b,1000,40)})
(102,{(102,c,d,1500,25)})
(103,{(103,e,f,2000,24)})
(104,{(104,g,h,1500,24)})
(105,{(105,i,j,3000,25)})

Week6:Retrieving user login credentials from /etc/passwd using Pig Latin

Steps:

1. Open your VM workstation and start the virtual machine you want to work with.

2. Open a terminal or command prompt within the VM.
3. Install Apache Pig on the VM. The installation steps may vary depending on the operating system running on your virtual machine.
4. Once Pig is installed, create a new Pig Latin script file (e.g., passwd.pig) and open it in a text editor.
5. In the Pig Latin script, use the PigStorage function to load the /etc/passwd file and split the fields using the colon delimiter. Then, filter out any lines starting with a hash (#) to exclude comments.
Here's an example Pig Latin script that accomplishes this:
Pig:

```
-- Load the /etc/passwd file
passwd = LOAD '/etc/passwd' USING PigStorage(':');

-- Filter out lines starting with a hash
filtered_passwd = FILTER passwd BY NOT(STARTSWITH($0, '#'));

-- Extract the username and password fields
user_credentials = FOREACH filtered_passwd GENERATE $0 AS username, $1 AS password;

-- Output the username and password
STORE user_credentials INTO 'output' USING PigStorage(',');
-- Load
```

Save the Pig Latin script.
6. In the terminal or command prompt within the VM, navigate to the directory where the passwd.pig script is located.
7. Run the Pig Latin script using the pig command:
Bash

```
pig passwd.pig
```

This will execute the Pig Latin script, loading the /etc/passwd file, filtering out comments, extracting the username and password fields, and storing the results in the output directory using a comma as the field delimiter.
8. After the script finishes running, you can access the output file generated by Pig (in this case, in the output directory) to view the retrieved user login credentials.
Please note that accessing system files such as /etc/passwd may require elevated privileges. Ensure that you have the necessary permissions or run the Pig script with appropriate privileges on the virtual machine.
As always, be cautious when dealing with sensitive user data and respect privacy regulations.

Week7: Sqoop installation and sqoop workouts

Step 1: Installing Sqoop

The following commands are used to extract the Sqoop tar ball and move it to "/usr/lib/sqoop" directory.

```
cd ~/install
tar xvzf sqoop-1.4.5.bin__hadoop-2.0.4-alpha.tar.gz
sudo mv sqoop-1.4.5.bin___hadoop-2.0.4-alpha /usr/local/sqoop
```

Step 2: Configuring bashrc

You have to set up the Sqoop environment by appending the following lines to

~/.bashrc file:vi ~/.bashrc

```
echo 'export SQOOP_HOME=/usr/local/sqoop' >>
~/.bashrc echo 'export
PATH=$PATH:$SQOOP_HOME/bin' >> ~/.bashrc
```

The following command is used to execute ~/.bashrc

file.source ~/.bashrc

Step 3: Configuring Sqoop

To configure Sqoop with Hadoop, you need to edit the sqoop-env.sh file, which is placed in the $SQOOP_HOME/conf directory. First of all, Redirect to Sqoop config directory and copy the template file using the following command:

```
cd $SQOOP_HOME/conf
mv sqoop-env-template.sh sqoop-env.sh
```

Open sqoop-env.sh and edit the following lines: (vi sqoop-env.sh)

```
echo 'export HADOOP_COMMON_HOME=/usr/local/hadoop' >>
sqoop-env.sh
echo 'export HADOOP_MAPRED_HOME=/usr/local/hadoop' >>
sqoop-env.sh
```

Step 4: Download and Configure mysql-connector-java

```
cd ~/install
cp -p mysql-connector-java.jar /usr/local/sqoop/
```

Step 5: Verifying Sqoop

The following command is used to verify the Sqoop version.source ~/.bashrc
sqoop-version

MYSQL (Preparation of Source) :

Login as mysql and start the Service :

su root
password :
hduser su
mysql
service mysqld
restartmysql

Select the test database:

use test;
CREATE TABLE customer (custid INT,firstname VARCHAR(20),lastname VARCHAR(20),city varchar(50),age int,createdt date,transactamt int);

insert into customer values(1,'Arun','Kumar','chennai',33,'2015-09-20',100000); insert into customer values(2,'srini','vasan','chennai',33,'2015-09-21',10000); insert into customer values(3,'vasu','devan','banglore',39,'2015-09-23',90000); insert into customer values(4,'mohamed','imran','hyderabad',33,'2015-09-24',1000); insert into customer values(5,'arun','basker','chennai',23,'2015-09-20',200000); insert into customer values(6,'ramesh','babu','manglore',39,'2015-09-21',100000);

select from customer;

SQOOP WORKOUTS (Open a separate linux terminal)

To List Databases which are in MySql

sqoop list-databases --connect jdbc:mysql://localhost --username root --password root;

To List Tables from test database

sqoop list-tables --connect jdbc:mysql://localhost/test --username root --password root;

Import Table from SQL to HDFS:

sqoop import --connect jdbc:mysql://localhost/test --username root --password root -table

customer -m 1 ; hadoop fs -rm -r /user/hduser/customer
sqoop import --connect jdbc:mysql://localhost/test --username root --password root -table customer -m 1 --
direct;

Import with –split-by option

sqoop import --connect jdbc:mysql://localhost/test --username root --password root -table customer -m 10 --split-by custid;
hadoop fs -rm -r /user/hduser/customer
sqoop import --connect jdbc:mysql://localhost/test --username root --password root -table customer -m 3 --split-by city;

Export from HDFS to SQL: (Create table in MYSQL before running this commmand)

CREATE TABLE customer1 (custid INT,firstname VARCHAR(20),lastname VARCHAR(20),city varchar(50),age int,createdt date,transactamt int);

sqoop export --connect jdbc:mysql://localhost/test --username root --password root --table customer1 --export-dir savedjob1

WEEK8: i) Hive installation
ii) Working with HiveQL

HIVE Installation with local metastore configuration
---

1) copy the hive tar file to your home path (/home/hduser/install) and extract using below commandcd /home/hduser/install/
tar xvzf apache-hive-0.14.0-bin.tar.gz

sudo mv apache-hive-0.14.0-bin /usr/local/hive

2) Make an entry in the bash profile like below,vi ~/.bashrc
export
HIVE_HOME=/usr/local/hive
export
PATH=$PATH:$HIVE_HO
ME/bin

Then source the bash profile

like below,source ~/.bashrc

3) After you complete the above steps execute below commands to create directories and give permissions

hadoop fs -mkdir -p /user/hive/warehouse/ hadoop fs -chmod g+w /user/hive/warehouse
hadoop fs -chmod g+w /tmp

4) Cd to bin folder inside

hive path cd

/usr/local/hive/bin

put an entry like

below, vi hive-

config.sh

export HADOOP_HOME=/usr/local/hadoop

==========

A. Create Database

---

create database retail;

B. Select Database

---

use retail;

set hive.cli.print.current.db=true;

C. Create table for storing transactional records

---

creating managed tables:

---

create table txnrecords(txnno INT, txndate STRING, custno INT, amount DOUBLE, category STRING, product STRING, city STRING, state STRING, spendby STRING) row format delimited
fields terminated by ',' lines terminated by '\n'
stored as textfile;

!mkdir -p /home/hduser/hive/data;

Copy all content from data path in the pen drive to the above path. /home/hduser/hive/data

Creating External tables:
---

create external table externaltxnrecords(txnno INT, txndate STRING, custno INT, amount DOUBLE, category STRING, product STRING, city STRING, state STRING, spendby STRING)
row format
delimited fields
terminated by ','
stored as textfile location '/user/hduser/hiveexternaldata';

D. Load the data into the table [From Linux client]

LOAD DATA LOCAL INPATH '/home/hduser/hive/data/txns' OVERWRITE

INTO TABLE txnrecords; Load the data into the table [From HDFS]
---

cd /home/hduser/hive/data/
hadoop fs -copyFromLocal
txns txns1

LOAD DATA INPATH '/user/hduser/txns1' OVERWRITE INTO

TABLE txnrecords; Select the loaded data

Map/reduce execution

check? select count(1)
from txnrecords ; select *
from txnrecords limit 10;
select * from txnrecords where category='Puzzles';

E. Describing metadata or schema of the table
---
describe formatted txnrecords;

describe formatted externaltxnrecord

Week-9

HiveQL - Functions

The Hive provides various in-built functions to perform mathematical and aggregate type operations. Here, we are going to execute such type of functions on the records of the below table:

employee_data

Id	Name	Salary
1	Gaurav	30000
2	Aryan	20000
3	Vishal	40000
4	John	10000
5	Henry	25000
6	William	9000
7	Lisa	25000
8	Ronit	20000

hive> create table employee_data (Id int, Name string , Salary float)
row format delimited
fields terminated by ',' ;

Now, load the data into the table.

Load:hive> load data local inpath '/home/codegyani/hive/emp_details' into table employee_data;

Let's fetch the loaded data by using the following command: -

hive> select * from employee_data;

```
hive> select * from employee_data;
OK
1       "Gaurav"        30000.0
2       "Aryan" 20000.0
3       "Vishal"        40000.0
4       "John"  10000.0
5       "Henry" 25000.0
6       "William"       9000.0
7       "Lisa"  25000.0
8       "Ronit" 20000.0
NULL    NULL    NULL
NULL    NULL    NULL
Time taken: 0.44 seconds, Fetched: 10 row(s)
hive>
```

Mathematical Functions in Hive

hive> select Id, Name, sqrt(Salary) from employee_data ;

```
hive> select Id, Name, sqrt(Salary) from employee_data ;
OK
1       "Gaurav"        173.20508075688772
2       "Aryan" 141.4213562373095
3       "Vishal"        200.0
4       "John"  100.0
5       "Henry" 158.11388300841898
6       "William"       94.86832980505137
7       "Lisa"  158.11388300841898
8       "Ronit" 141.4213562373095
NULL    NULL    NULL
NULL    NULL    NULL
Time taken: 0.754 seconds, Fetched: 10 row(s)
hive>
```

Aggregate Functions in Hive

In Hive, the aggregate function returns a single value resulting from computation over many rows. Let"s see some commonly used aggregate functions: -

hive> select max(Salary) from employee_data;

```
Total MapReduce CPU Time Spent: 23 seconds 840 msec
OK
40000.0
Time taken: 231.973 seconds, Fetched: 1 row(s)
hive>
```

hive> select min(Salary) from employee_data;

```
codegyani@ubuntu64server: ~/hive
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2019-08-27 11:49:45,475 Stage-1 map = 0%,  reduce = 0%
2019-08-27 11:50:45,943 Stage-1 map = 0%,  reduce = 0%
2019-08-27 11:51:09,936 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 11.52 s
ec
2019-08-27 11:52:10,219 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 11.52 s
ec
2019-08-27 11:52:21,631 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 14.71
sec
2019-08-27 11:52:40,408 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 24.02
 sec
MapReduce Total cumulative CPU time: 24 seconds 20 msec
Ended Job = job_1555046592674_0057
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 24.02 sec   HDFS Read: 7112 H
DFS Write: 7 SUCCESS
```

```
codegyani@ubuntu64server: ~/hive
Total MapReduce CPU Time Spent: 24 seconds 20 msec
OK
9000.0
Time taken: 275.472 seconds, Fetched: 1 row(s)
hive>
```

Other built-in Functions in Hive:

select Id, upper(Name) from employee_data;

```
codegyani@ubuntu64server: ~/hive
hive> select Id, upper(Name) from employee_data;
OK
1       "GAURAV"
2       "ARYAN"
3       "VISHAL"
4       "JOHN"
5       "HENRY"
6       "WILLIAM"
7       "LISA"
8       "RONIT"
NULL    NULL
NULL    NULL
Time taken: 0.529 seconds, Fetched: 10 row(s)
hive>
```

select Id, lower(Name) from employee_data;

```
codegyani@ubuntu64server: ~/hive
hive> select Id, lower(Name) from employee_data;
OK
1       "gaurav"
2       "aryan"
3       "vishal"
4       "john"
5       "henry"
6       "william"
7       "lisa"
8       "ronit"
NULL    NULL
NULL    NULL
Time taken: 0.516 seconds, Fetched: 10 row(s)
hive>
```

10.Perform classification & clustering in Mahout Hadoop,Building a Mahout Recommendation System on a Hadoop Cluster

Mahout - Clustering

Clustering is the procedure to organize elements or items of a given collection into groups based on the similarity between the items. For example, the applications related to online news publishing group their news articles using clustering.

Applications of Clustering

- Clustering is broadly used in many applications such as market research, pattern recognition, data analysis, and image processing.
- Clustering can help marketers discover distinct groups in their customer basis. And they can characterize their customer groups based on purchasing patterns.
- In the field of biology, it can be used to derive plant and animal taxonomies, categorize genes with similar functionality and gain insight into structures inherent in populations.
- Clustering helps in identification of areas of similar land use in an earth observation database.
- Clustering also helps in classifying documents on the web for information discovery.
- Clustering is used in outlier detection applications such as detection of credit card fraud.
- As a data mining function, Cluster Analysis serves as a tool to gain insight into the distribution of data to observe characteristics of each cluster.

Using Mahout, we can cluster a given set of data. The steps required are as follows:

- **Algorithm** You need to select a suitable clustering algorithm to group the elements of a cluster.
- **Similarity and Dissimilarity** You need to have a rule in place to verify the similarity between the newly encountered elements and the elements in the groups.
- **Stopping Condition** A stopping condition is required to define the point where no clustering is required.

Procedure of Clustering

To cluster the given data you need to -

- Start the Hadoop server. Create required directories for storing files in Hadoop File System. (Create directories for input file, sequence file, and clustered output in case of canopy).
- Copy the input file to the Hadoop File system from Unix file system.
- Prepare the sequence file from the input data.
- Run any of the available clustering algorithms.
- Get the clustered data.

Starting Hadoop

Mahout works with Hadoop, hence make sure that the Hadoop server is up and running.

```
$ cd HADOOP_HOME/bin
$ start-all.sh
```

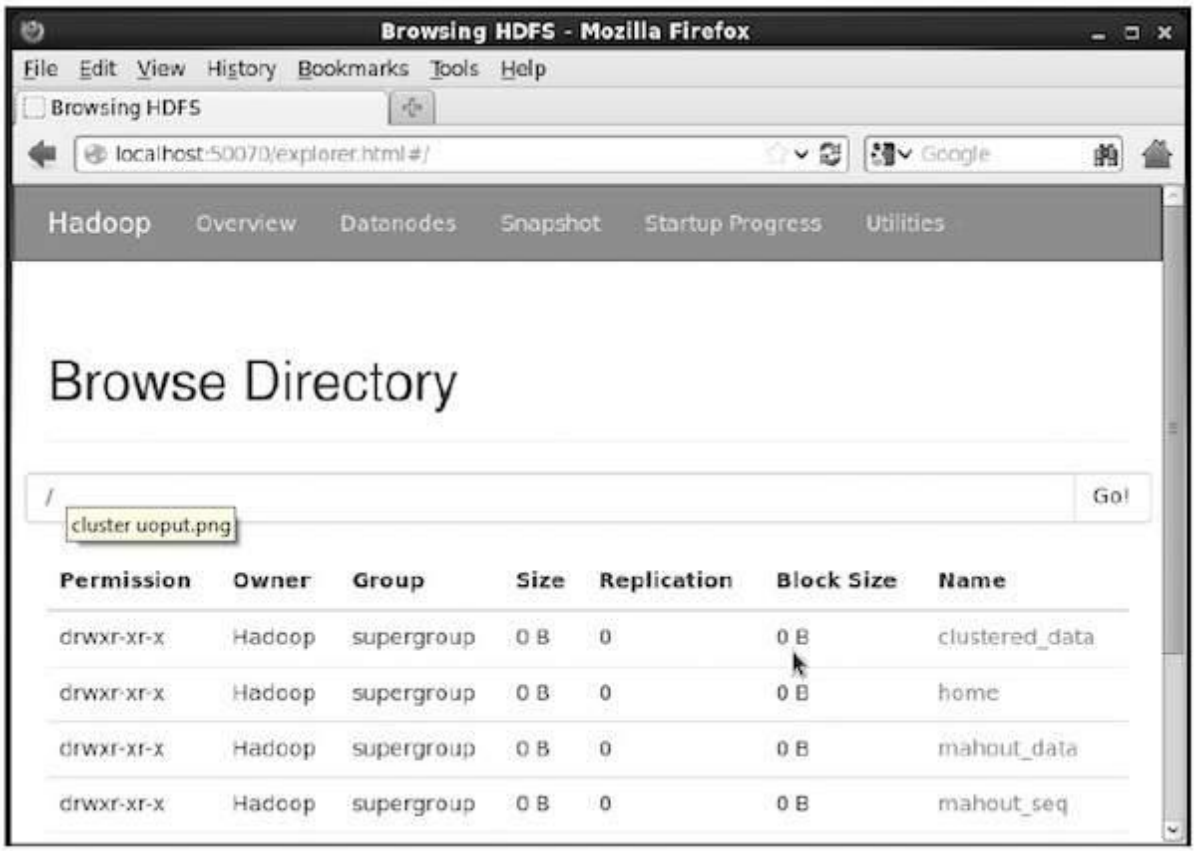
Preparing Input File Directories

Create directories in the Hadoop file system to store the input file, sequence files, and clustered data using the following command:

```
$ hadoop fs -p mkdir /mahout_data
$ hadoop fs -p mkdir /clustered_data
$ hadoop fs -p mkdir /mahout_seq
```

You can verify whether the directory is created using the hadoop web interface in the following URL - **http://localhost:50070/**

It gives you the output as shown below:



Copying Input File to HDFS

Now, copy the input data file from the Linux file system to mahout_data directory in the Hadoop File System as shown below. Assume your input file is mydata.txt and it is in the /home/Hadoop/data/ directory.

$ hadoop fs -put /home/Hadoop/data/mydata.txt /mahout_data/

Preparing the Sequence File

Mahout provides you a utility to convert the given input file in to a sequence file format. This utility requires two parameters.

- The input file directory where the original data resides.
- The output file directory where the clustered data is to be stored.

Given below is the help prompt of mahout **seqdirectory** utility.

Step 1: Browse to the Mahout home directory. You can get help of the utility as shown below:

```
[Hadoop@localhost bin]$ ./mahout seqdirectory --help
Job-Specific Options:
--input (-i) input Path to job input directory.
--output (-o) output The directory pathname for output.
--overwrite (-ow) If present, overwrite the output directory
```

Generate the sequence file using the utility using the following syntax:

mahout seqdirectory -i <input file path> -o <output directory>

Example

```
mahout seqdirectory
-i hdfs://localhost:9000/mahout_seq/
-o hdfs://localhost:9000/clustered_data/
```

Building a Recommender using Mahout

Here are the steps to develop a simple recommender:

Step1: Create DataModel Object

The constructor of **PearsonCorrelationSimilarity** class requires a data model object, which holds a file that contains the Users, Items, and Preferences details of a product. Here is the sample data model file:

```
1,00,1.0
1,01,2.0
1,02,5.0
1,03,5.0
1,04,5.0

2,00,1.0
2,01,2.0
2,05,5.0
2,06,4.5
2,02,5.0
```

3,01,2.5
3,02,5.0
3,03,4.0
3,04,3.0

4,00,5.0
4,01,5.0
4,02,5.0
4,03,0.0

The **DataModel** object requires the file object, which contains the path of the input file. Create the **DataModel** object as shown below.

DataModel datamodel = new FileDataModel(new File("input file"));
Step2: Create UserSimilarity Object

Create **UserSimilarity** object using **PearsonCorrelationSimilarity** class as shown below:

UserSimilarity similarity = new PearsonCorrelationSimilarity(datamodel);
Step3: Create UserNeighborhood object

This object computes a "neighborhood" of users like a given user. There are two types of neighborhoods:

- **NearestNUserNeighborhood** - This class computes a neighborhood consisting of the nearest *n* users to a given user. "Nearest" is defined by the given UserSimilarity.
- **ThresholdUserNeighborhood** - This class computes a neighborhood consisting of all the users whose similarity to the given user meets or exceeds a certain threshold. Similarity is defined by the given UserSimilarity.

Here we are using **ThresholdUserNeighborhood** and set the limit of preference to 3.0.

UserNeighborhood neighborhood = new ThresholdUserNeighborhood(3.0, similarity, model);
Step4: Create Recommender Object

Create **UserbasedRecomender** object. Pass all the above created objects to its constructor as shown below.

UserBasedRecommender recommender = new GenericUserBasedRecommender(model, neighborhood, similarity);
Step5: Recommend Items to a User

Recommend products to a user using the recommend() method of **Recommender** interface. This method requires two parameters. The first represents the user id of the user to whom we need to send the recommendations, and the second represents the number of recommendations to be sent. Here is the usage of **recommender()** method:

List<RecommendedItem> recommendations = recommender.recommend(2, 3);

for (RecommendedItem recommendation : recommendations) {

```
  System.out.println(recommendation);
 }
```

Example Program

Given below is an example program to set recommendation. Prepare the recommendations for the user with user id 2.

```
import java.io.File;
import java.util.List;

import org.apache.mahout.cf.taste.impl.model.file.FileDataModel;
import org.apache.mahout.cf.taste.impl.neighborhood.ThresholdUserNeighborhood;
import org.apache.mahout.cf.taste.impl.recommender.GenericUserBasedRecommender;
import org.apache.mahout.cf.taste.impl.similarity.PearsonCorrelationSimilarity;

import org.apache.mahout.cf.taste.model.DataModel;
import org.apache.mahout.cf.taste.neighborhood.UserNeighborhood;

import org.apache.mahout.cf.taste.recommender.RecommendedItem;
import org.apache.mahout.cf.taste.recommender.UserBasedRecommender;

import org.apache.mahout.cf.taste.similarity.UserSimilarity;

public class Recommender {
  public static void main(String args[]){
    try{
      //Creating data model
      DataModel datamodel = new FileDataModel(new File("data")); //data

      //Creating UserSimilarity object.
      UserSimilarity usersimilarity = new PearsonCorrelationSimilarity(datamodel);

      //Creating UserNeighbourHHood object.
      UserNeighborhood userneighborhood = new ThresholdUserNeighborhood(3.0,
usersimilarity, datamodel);

      //Create UserRecomender
      UserBasedRecommender recommender = new
GenericUserBasedRecommender(datamodel, userneighborhood, usersimilarity);

      List<RecommendedItem> recommendations = recommender.recommend(2, 3);

      for (RecommendedItem recommendation : recommendations) {
        System.out.println(recommendation);
      }
```

```
    }catch(Exception e){}

  }
}
```

Compile the program using the following commands:

javac Recommender.java
java Recommender

It should produce the following output:

RecommendedItem [item:3, value:4.5]
RecommendedItem [item:4, value:4.0]

VIVA VOCE

BIG DATA ANALYTICS VIVA QUESTIONS

1. What is the full form of HDFS?

Answer: Hadoop Distributed File System.

2. How can you list files in HDFS using a command?

Answer: hadoop fs -ls.

3. Explain the role of Mapper in Hadoop's MapReduce framework.

Answer: The Mapper processes input data and generates key-value pairs.

4. What does a Reducer do in MapReduce?

Answer: The Reducer aggregates and summarizes data from the Mapper.

5. Why use a multi-node cluster for analyzing datasets?

Answer: Multi-node clusters speed up analysis by distributing tasks across nodes.

6. Which HDFS command is used to create a directory?

Answer: hadoop fs -mkdir.

7. What is a User Defined Function (UDF) in Pig used for?

Answer: UDFs perform customized data processing tasks in Pig scripts.

8. How does Hadoop ensure fault tolerance for data?

Answer: Hadoop replicates data blocks across nodes.

9. How can you copy a file from local storage to HDFS?

Answer: hadoop fs -put.

10. Explain the purpose of Hive in Hadoop.

Answer: Hive provides SQL-like querying for Hadoop data.

11. How does the Word Count application work in Hadoop?

Answer: It counts the frequency of words in a dataset.

12. What is the default replication factor in HDFS?

Answer: 3.

13. What is the purpose of the YARN ResourceManager?

Answer: It manages resources and scheduling in Hadoop.

14. What is the output of the Reducer in Word Count?

Answer: Aggregated word counts.

15. How does data locality enhance Hadoop performance?

Answer: It minimizes data movement over the network.

16. How can you copy a file within HDFS using a command?

Answer: hadoop fs -cp.

17. Explain the purpose of Secondary NameNode in HDFS.

Answer: It helps in periodic metadata checkpointing.

18. What is the significance of HiveQL in Hadoop?

Answer: HiveQL provides SQL-like querying for Hive.

19. How is data divided for parallel processing in HDFS?

Answer: Data is divided into fixed-size blocks.

20. What is the purpose of JobTracker in Hadoop?

Answer: It manages job scheduling in MapReduce.

21. How can you read the first few lines of a file in HDFS using a command?

Answer: hadoop fs -tail.

22. What is the goal of Mahout's Recommendation System?

Answer: To provide personalized suggestions.

23. How does HDFS handle data replication?

Answer: It replicates data blocks on multiple nodes.

24. How can you execute a Pig script?

Answer: pig script_name.pig.

25. What is the significance of data partitioning in Hive?

Answer: It improves query performance.

26. How can you remove a file from HDFS?

Answer: hadoop fs -rm.

27. Explain the concept of MapReduce in Hadoop.

Answer: MapReduce is a programming model for processing large datasets in parallel.

28. How is Hive different from traditional RDBMS?

Answer: Hive operates on structured data stored in Hadoop, whereas RDBMS works with relational databases.

29. What is the purpose of TaskTracker in Hadoop?

Answer: It executes tasks assigned by the JobTracker.

30. How can you create a table in Hive using HiveQL?

Answer: Use the CREATE TABLE statement.

31. What does the hadoop fs -get command do?

Answer: It copies files from HDFS to the local file system.

32. What is the purpose of the Mapper component in Word Count?

Answer: It tokenizes and generates intermediate key-value pairs.

33. What does the YARN ResourceManager manage?

Answer: Resources and applications in the cluster.

34. How does Hive optimize query performance?

Answer: It generates query execution plans.

35. How can you define a Pig UDF using Python?

Answer: Write a Python script and register it in the Pig script.

36. What is the role of the TaskTracker in Hadoop?

Answer: The TaskTracker runs MapReduce tasks.

37. What is the purpose of the hadoop fs -tail command?

Answer: It displays the end of a file in HDFS.

38. What is the main function of the Mahout clustering algorithm?

Answer: To group similar data points.

39. How can you filter data in HiveQL?

Answer: Use the WHERE clause.

40. How do you delete a directory in HDFS using a command?

Answer: Use hadoop fs -rm -r.

41. What is the purpose of Hadoop's Secondary NameNode?

Answer: To assist the NameNode in checkpointing metadata.

42. Which HDFS command is used to copy files from HDFS to the local file system?

Answer: hadoop fs -get.

43. Explain the role of a Reducer in Hadoop.

Answer: The Reducer processes and aggregates intermediate data.

44. How does Hive optimize query execution?

Answer: It generates optimal query plans.

45. What is data replication in HDFS and why is it important?

Answer: Data replication is duplicating data blocks on different nodes for fault tolerance.

46. How can you remove a directory from HDFS?

Answer: hadoop fs -rm -r.

47. What is the use of the hadoop fs -tail command?

Answer: It displays the end of a file in HDFS.

48. How does Mahout's classification work?

Answer: Mahout's classification assigns data to predefined categories.

49. How can you write a Hive UDF using Java?

Answer: Create a Java class extending org.apache.hadoop.hive.ql.UDF.

50. What is the purpose of the TaskTracker in Hadoop's MapReduce?

Answer: The TaskTracker executes individual tasks in MapReduce jobs.

51. Which HDFS command is used to copy files within HDFS?

Answer: hadoop fs -cp.

52. What is the core principle of a Recommendation System built with Mahout?

Answer: Providing personalized suggestions based on user behavior.

53. How is data distributed across nodes in HDFS?

Answer: Data is divided into fixed-size blocks and distributed across nodes.

54. How can you read the contents of a file in HDFS using a terminal command?

Answer: Use hadoop fs -cat.

55. Explain the role of the YARN ResourceManager in a Hadoop cluster.

Answer: The YARN ResourceManager allocates resources to applications and manages cluster resources.

56. What is the purpose of Hive in the Hadoop ecosystem?

Answer: Hive provides a SQL-like interface for querying and managing data stored in Hadoop.

57. How does the NameNode ensure data integrity in HDFS?

Answer: The NameNode maintains the metadata and block locations for data consistency and replication.

58. How can you create a new directory in HDFS using a command?

Answer: Use hadoop fs -mkdir.

59. What is the primary goal of Mahout's clustering algorithms?

Answer: To group similar data points together.

60. How can you execute a Pig script from the command line?

Answer: Use the command pig script_name.pig.

61. What is the purpose of the Secondary NameNode in HDFS?

Answer: The Secondary NameNode assists the NameNode by periodically checkpointing metadata.

62. Which HDFS command is used to delete a file?

Answer: hadoop fs -rm.

63. Explain the working of the Mapper in Hadoop's MapReduce framework.

Answer: The Mapper processes input data and generates intermediate key-value pairs.

64. How does data replication contribute to HDFS reliability?

Answer: Data replication ensures fault tolerance by storing multiple copies of data blocks.

65. How can you copy files from HDFS to the local file system using a command?

Answer: Use hadoop fs -get.

66. What is the purpose of the Hive Query Language (HiveQL)?

Answer: HiveQL is used for querying and managing structured data in Hadoop.

67. How does Hadoop's data locality optimization work?

Answer: Hadoop schedules tasks on nodes that have the required data, reducing network traffic.

68. How can you display the first few lines of a file in HDFS using a command?

Answer: Use hadoop fs -cat.

69. Explain the process of classification in Mahout Hadoop.

Answer: Classification in Mahout involves training a model to assign data to predefined categories.

70. What is the significance of partitions in Hive tables?

Answer: Partitions enable efficient data organization and querying.

71. How can you copy a directory from local storage to HDFS?

Answer: Use hadoop fs -put.

72. What role does the ResourceManager play in Hadoop's YARN?

Answer: The ResourceManager allocates resources to applications in the cluster.

73. How can you create a table in Hive using HiveQL?

Answer: Use the CREATE TABLE statement in HiveQL.

74. What does the hadoop fs -get command do?

Answer: It copies files from HDFS to the local file system.

75. Explain the function of the Mapper in Hadoop.

Answer: The Mapper processes input data and produces intermediate key-value pairs.

76.How does the Reducer component contribute to the MapReduce process?

Answer: The Reducer aggregates and summarizes the intermediate results generated by the Mapper.

77. How does a multi-node cluster enhance data processing performance in Hadoop?

Answer: Multi-node clusters distribute the processing load, leading to faster data analysis.

78. What is the purpose of the hadoop fs -rm command?

Answer: It is used to remove files from HDFS.

79. What is the function of User Defined Functions (UDFs) in Pig?

Answer: UDFs allow customized data processing in Pig scripts.

80. How is data replication achieved in HDFS?

Answer: Data replication involves storing multiple copies of data blocks on different nodes.

81. How can you copy a file within HDFS using a command?

Answer: Use hadoop fs -cp.

82. What is HiveQL used for in the context of Hive?

Answer: HiveQL is a query language for querying and managing data stored in Hive.

83. How does Hive optimize query execution?

Answer: Hive optimizes queries by generating efficient query execution plans.

84. What is the purpose of the Secondary NameNode in HDFS?

Answer: The Secondary NameNode assists the NameNode in checkpointing metadata for recovery.

85. How can you read the end of a file in HDFS using a command?

Answer: Use hadoop fs -tail.

86. How does Mahout's clustering algorithm work?

Answer: Mahout's clustering algorithm groups data points with similar characteristics.

87. How can you define a Hive User Defined Function (UDF) using Java?

Answer: Define a Java class extending org.apache.hadoop.hive.ql.UDF.

88. What is the purpose of the TaskTracker in Hadoop's MapReduce?

Answer: The TaskTracker manages task execution on worker nodes.

89. Which HDFS command is used to copy files from HDFS to the local file system?

Answer: hadoop fs -get.

90. Explain the function of the Reducer in MapReduce.

Answer: The Reducer aggregates and summarizes data from the Mapper phase.

91. How does Hive optimize query execution in Hadoop?

Answer: Hive generates optimized execution plans for queries.

92. What is the role of data replication in HDFS?

Answer: Data replication provides fault tolerance and data availability.

93. How can you copy a file from the local file system to HDFS?

Answer: Use hadoop fs -put.

94. What is the purpose of Hive in the Hadoop ecosystem?

Answer: Hive provides SQL-like querying for structured data in Hadoop.

95. How does the Word Count application work in Hadoop?

Answer: It processes text data to count word occurrences.

96. How does Hive optimize query performance?

Answer: Hive optimizes performance by generating efficient execution plans.

97. How does data locality enhance Hadoop's efficiency?

Answer: Data locality reduces data movement across the network.

98. How can you copy a file within HDFS using a command?

Answer: Use hadoop fs -cp.

99. What is the role of the Hive Query Language (HiveQL) in Hadoop?

Answer: HiveQL is used to query and manage data stored in Hive.

100. What is the purpose of Mahout's Recommendation System in a Hadoop cluster?

Answer: Mahout's Recommendation System provides personalized suggestions based on user preferences.