**VELAGAPUDI RAMAKRISHNA SIDDHARTHA ENGINEERING COLLEGE::VIJAYAWADA**

**(AUTONOMOUS)**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**LABORATORY MANUAL**

**Course Code: 17IT4651A**

**Course Name: BIG DATA LAB**

**Year/Semester: III/IV – II Semester**

**Academic Year: 2021-2022**

**Prepared By:**

1. **Dr. Shaik Fathimabi**
2. **Ch. Nanda Krishna**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| SNo | Task | Page No |
| 1. | Syllabus Content | 3 |
| 2. | CO’s, PO’s and PSO’s   * 1. Course Outcomes, Program Outcomes/Program Specific Outcomes   2. CO-PO-PSO Mapping   3. List of Experiments with CO-PO-PSO mapping and Blooms Taxonomy level | 5 |
| 3. | Solutions w.r.t experiments | 7 |
| 4. | Internal and External Question Papers with CO-PO mapping | 53 |
| 5. | Internal and External Question Paper Analysis  (course outcome wise and Blooms taxonomy level wise) | 62 |
| 6. | Project / Case Studies List | 63 |
| 7. | Experiments beyond the syllabus / projects for Advanced  Learners | 72 |
| 8. | Viva-Voce Questions | 73 |
| 9. | Rubric for Internal Evaluation | 76 |
| 10. | Internal Marks Statement | 77 |
| 11. | Rubric for External Evaluation | 80 |
| 12. | External Marks Statement | 81 |
| 13. | FCAR | 81 |
| 14. | Course Summary | 85 |

**17IT4651A- BIG DATA LAB**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Course Category:** | | | Program Elective - II | | | | | | | **Credits:** | | | | | | | 1 | | |
| **Course Type:** | | | Lab | | | | | | | **Lecture-Tutorial-Practice:** | | | | | | | 0-0-2 | | |
| **Prerequisites:** | | | 17IT3402 - DBMS,  17IT3502 -Data Mining | | | | | | | **Continuous Evaluation:** | | | | | | | 30 | | |
|  | | | | | | | | | | **Semester end Evaluation:** | | | | | | | 70 | | |
| **Total Marks:** | | | | | | | 100 | | |
| **Course Outcomes** | Upon successful completion of the course, the student will be able to: | | | | | | | | | | | | | | | | | | |
| CO1 | | | Understand the concepts and challenges in analyzing big data. | | | | | | | | | | | | | | | |
| CO2 | | | Learn to work with ecosystems available in Hadoop. | | | | | | | | | | | | | | | |
| CO3 | | | Understand the impact of big data for business strategies & decisions. | | | | | | | | | | | | | | | |
| **Contribution of Course Outcomes towards achievement of Program Outcomes**  **(1-Low, 2-Medium, 3-High)** |  | PO1 | | | PO2 | PO3 | PO4 | PO5 | PO6 | | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | | PSO1 | PSO2 | |
| CO1 | 3 | | | 3 |  | 1 | 3 |  | |  |  | 1 |  |  |  | | 1 | 2 | |
| CO2 | 3 | | | 3 |  | 1 | 3 |  | |  |  | 1 |  |  |  | | 1 | 2 | |
| CO3 | 3 | | |  |  |  | 1 |  | |  |  | 1 |  |  | 1 | | 1 | 1 | |
| **Course Content** | **Week 1**   * Introduction to big Data * Applications of Big Data * Challenges of Big Data * Characteristics of Big Data * Tools | | | | | | | | | | | | | | | | | | |
| **Week 2**  MySQL Queries. | | | | | | | | | | | | | | | | | | |
| **Week 3**  Installation of Cloudera. | | | | | | | | | | | | | | | | | | |
| **Week 4**  Exploring HDFS. Listing of files, exploring dictionaries. | | | | | | | | | | | | | | | | | | |
|  | **Week 5**  Hdfs Operations using various commands. | | | | | | | | | | | | | | | | | | |
| **Week 6**  Hive architecture, Creating hive tables using hiveql language. | | | | | | | | | | | | | | | | | | |
| **Week 7**  Loading data into Hive warehouse. Apply aggregate operations on data. | | | | | | | | | | | | | | | | | | |
| **Week 8**  Implement partitioning of data in Hive Warehouse using HiveQL. | | | | | | | | | | | | | | | | | | |
| **Week 9**  Implement the concepts of pig. | | | | | | | | | | | | | | | | | | |
| **Week 10**  Implement the concept of map reduce for various examples. | | | | | | | | | | | | | | | | | | |
| **Week 11**  Case Study on Hive. | | | | | | | | | | | | | | | | | | |
| **Week 12**  Case Study on Map Reduce. | | | | | | | | | | | | | | | | | | |
| Text books and reference books | **Text Book(s)**  [1].Dirk deRoos, Chris Eaton, George Lapis, Paul Zikopoulos, Tom Deutsch,“Understanding Big Data Analytics for Enterprise Class Hadoop and Streaming Data”,1st Edition, TMH,2012.  [2].Tom White, Hadoop, “The Definitive Guide”, 3rd Edition, O’Reilly Publications, 2012.  **Reference Book(s)**  [1].Michael Berthold, David J. Hand, “Intelligent Data Analysis”, Springer, 2007.  [2].David Loshin, "BigDataAnalytics: From Strategic Planning to Enterprise Integration with Tools, Techniques, NoSQL, and Graph”,Morgan Kaufmann Publishers, 2013.  [3].Hadoop in Practice by Alex Holmes, MANNING Publ.  [4].Hadoop in Action by Chuck Lam, MANNING Publ. | | | | | | | | | | | | | | | | | | |
| E-resources and other digital material | [1].An overview of “Big Data”: Available <http://www.jbonneau.com/doc/2012-04-27-big_data_lecture_1.pdf>  [2].Hadoop Tutorial: Developing Big-Data Applications with Apache Hadoop: Available <http://www.coreservlets.com/hadoop-tutorial/>  [3].Random notes on big data – SlideShare: Available [www.slideshare.net/yiranpang/random-notes-on-big-data-26439474](http://www.slideshare.net/yiranpang/random-notes-on-big-data-26439474)  [4].http://www.cloudera.com/content/cloudera-content/clouderadocs/HadoopTutorial/CDH4/Hadoop -Tutorial.html  [5]. <https://www.ibm.com/developerworks/community/blogs> Susan Visser Editionntry/flash book understanding big data analytics for enterprise class hadoop and streaming data? lang en | | | | | | | | | | | | | | | | | | |

**Course Outcomes**

|  |  |
| --- | --- |
| CO1 | Understand the concepts and challenges in analyzing big data. |
| CO2 | Learn to work with ecosystems available in Hadoop. |
| CO3 | Understand the impact of big data for business strategies & decisions. |

## Program Outcomes

**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 modeling 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

**PSO1 :** Apply the concepts of Data Science, Software Modeling and Networking for IT applications

**PSO2 :** Discover mechanisms that would perform tasks related to Research, Education, Training and/or E- governance

**17IT4651A :: BIGDATA Lab**

**CO - PO Mapping**

The course outcomes of each core course are mapped to the Program Outcomes with numbers 1 , 2 or 3 a level of emphasis being either strongly correlated 3, medium correlated 2, moderately correlated 1. The level of emphasis of a program outcome is determined by the weight used for assessing the outcome in each course. The level of emphasis for an outcome is determined by the weight as follows:

When the course outcome weight is < 40%, it will be given as moderately correlated (1).

When the course outcome weight is >40% and <60 it will be given as medium correlated (2).

When the course outcome weight is >60%, it will be given as strongly correlated (3).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PO  1 | PO  2 | PO  3 | PO  4 | PO  5 | PO  6 | PO  7 | PO  8 | PO  9 | PO  10 | PO  11 | PO12 | PSO1 | PSO2 |
| CO1: Understand the concepts and challenges in analyzing big data. | 3 | 3 | - | 1 | 3 | - | - | - | 1 | - | - | - | 1 | 2 |
| CO2: Learn to work with ecosystems available in Hadoop. | 3 | 3 | - | 1 | 3 | - | - | - | 1 | - | - | - | 1 | 2 |
| CO3: Understand the impact of big data for business strategies & decisions. | 3 | - | - | - | 1 | - | - | - | 1 | - | - | 1 | - | 1 |

**III/IV B. Tech. VI Semester**

**AY::2021-2022**

**17IT4651A BIG DATA LAB**

**List of Experiments**

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment No** | **Description** | **Course Outcome** | **BTL** |
| **1.** | **Installation and Monitoring of Hadoop.** | **CO2** | **K2** |
| **2.** | **Exploring various Hadoop Distributed File System commands.** | **CO1** | **K3** |
| **3** | **Implementation of Cat command and file write operation on HDFS.** | **CO2** | **K3** |
| **4** | **Execution of mapreduce programs like wordcount, tera data generator and tera data sort using Hadoop Examples jar.** | **CO2** | **K4** |
| **5.** | **Implementation of MapReduce based Word count using python.** | CO2 | **K3** |
| **6** | **Implementation of Word count using Mapreduce with combiner.** | **CO1** | **K4** |
| 7 | **Implementation of MapReduce program that mines weather data.** | **CO3** | **K5** |
| **8** | **Implementation of data processing operators using Pig Latin.** | **CO2** | **K3** |
| **9** | **Implementation of data analytics using Pig.** | **CO2** | **K3** |
| **10** | **Creating Hive tables and loading data using Hive Query language.** | **CO2** | **K5** |
| **11** | **Implement partitioning of data in Hive Warehouse using Hive Query Language.** | **CO2** | **K5** |
| **12** | **Loading data into HIVE warehouse. Apply aggregate operations on data.** | **CO3** | K5 |

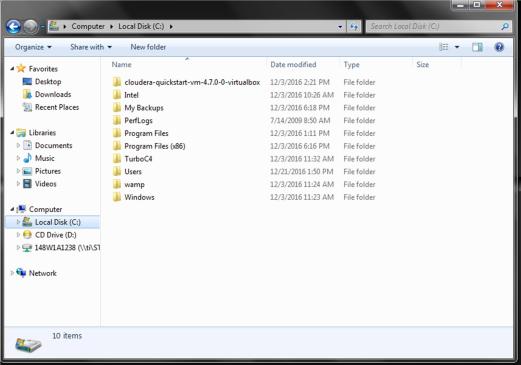
**Task-1**

**Aim: Installation and Monitoring of Hadoop.**

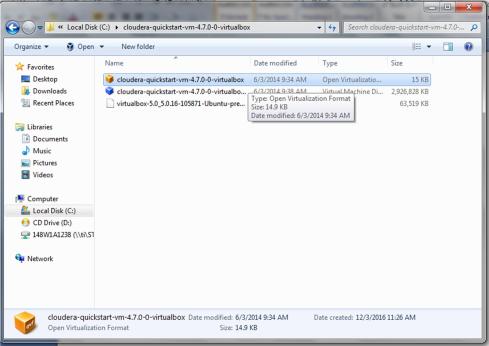
**Prerequisites**

* VIRTUAL BOX: it is used for installing the operating system on it.
* OPERATING SYSTEM: You can install Hadoop on Linux based operating systems. Ubuntu and CentOS are very commonly used. In this tutorial, we are using CentOS.
* JAVA: You need to install the Java 8 package on your system.
* HADOOP: You require Hadoop 2.7.3 package.

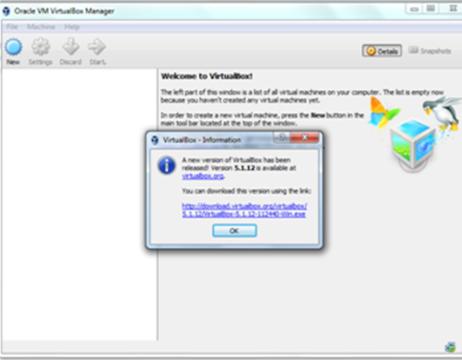
**Step1: Open c drive and click on cloudera-quickstart-vm-4.7.0-0-virtualbox**



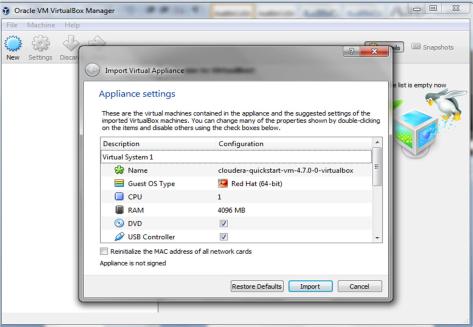
**Step2: Now click on cloudera-quickstart-vm-4.7.0-0-virtualbox**



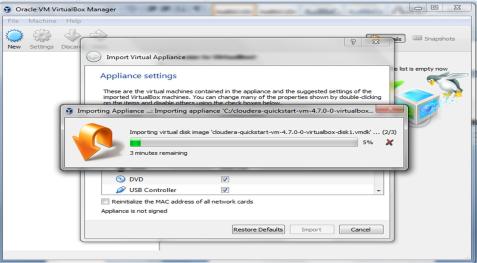
**Step3:After clicking on cloudera-quickstart-vm-4.7.0-0-virtualbox we will get a dialog box and after that we have click ok**



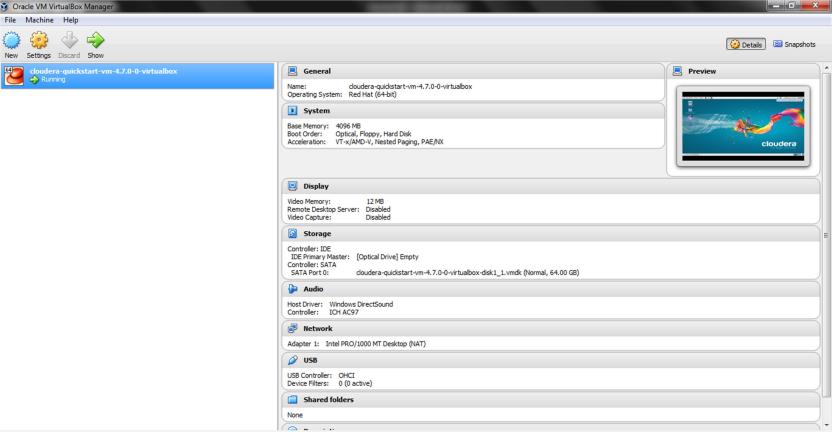
**Step4: Now click import to import the virtual machine**



**Step5: Now virtual machine is importing.**



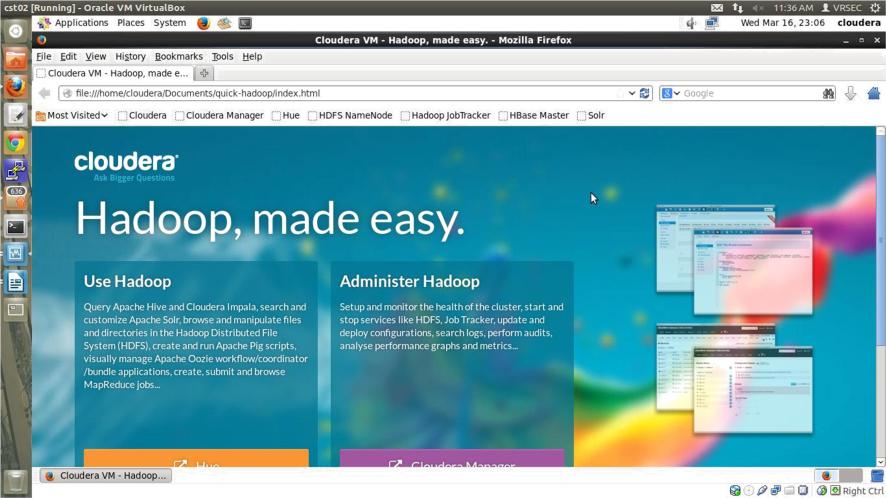
**Step6: After importing this dialog box appears and now click on start**

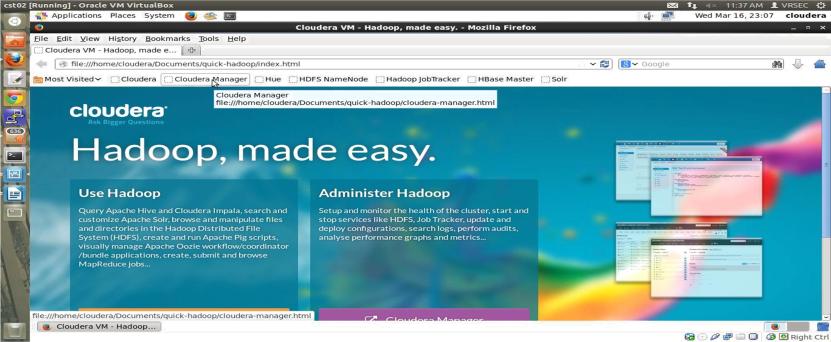


**Step7: Now virtual machine desktop is opened and it is as follows**

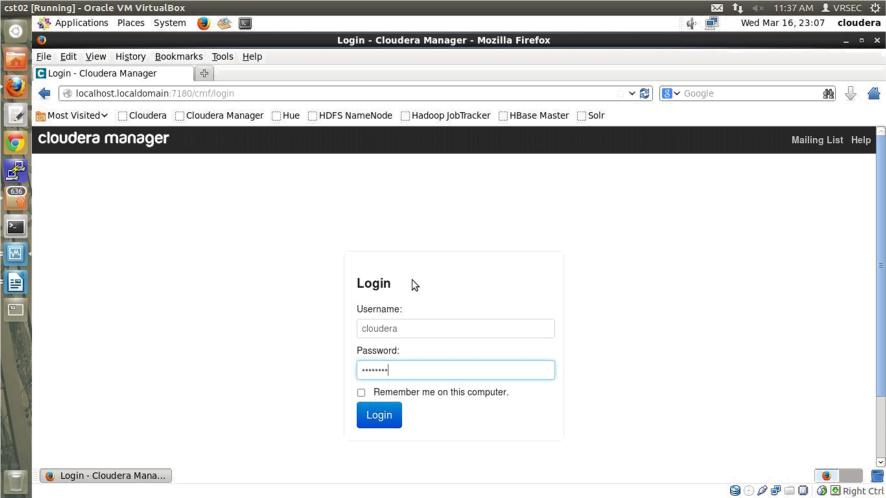
Exploring HDFS (Hadoop distributed file system) file system. Listing of files, exploring directories.

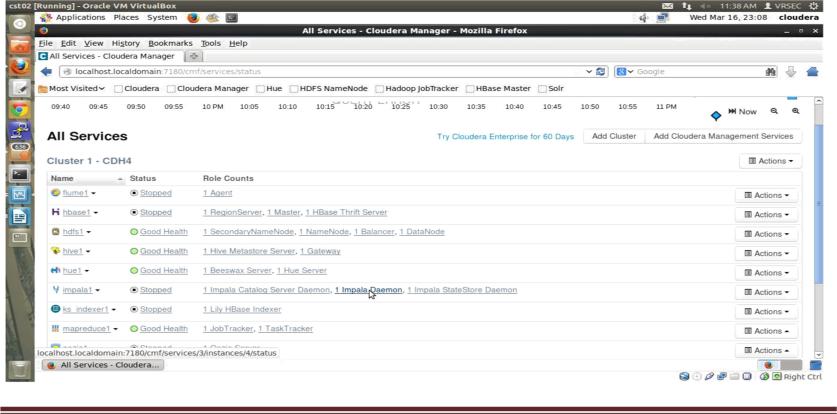
**Step1:** Open Virtual box tool->start services ,it displays cloudera homepage

**Step2:** Run Mozilla firebox browser to open index file.html page**.** Then click on cloudera manager, cloudera services page will be displayed.

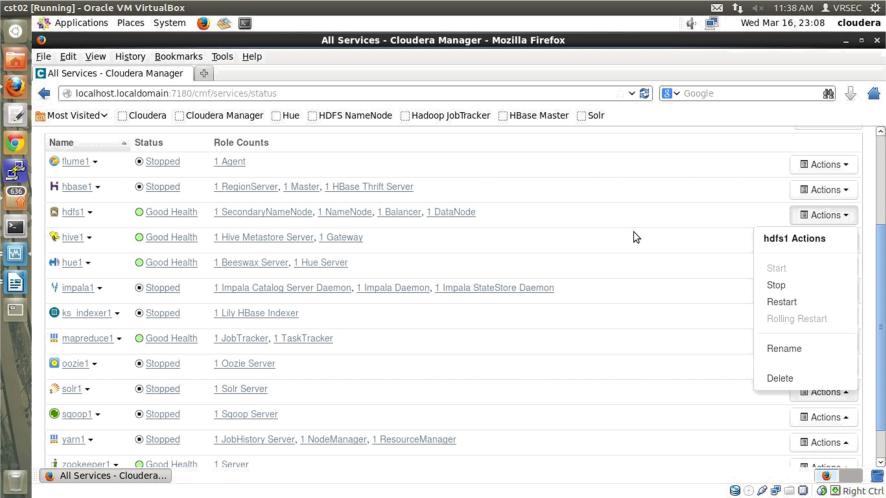


**Step3:** After selecting cloudera Manager, Provide Authentication Details Username : cloudera password: cloudera

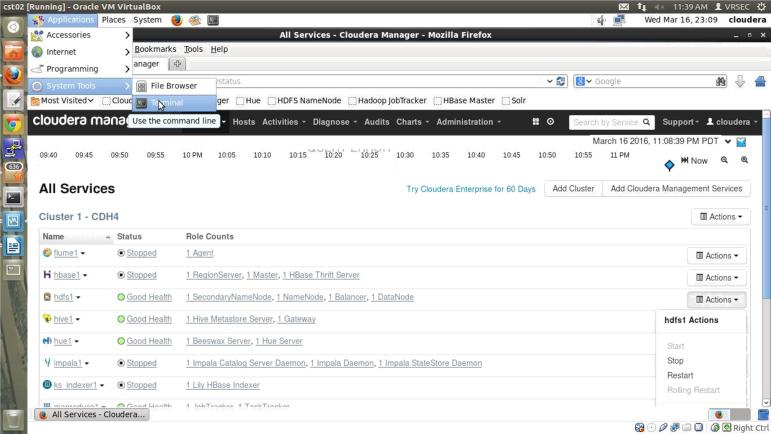


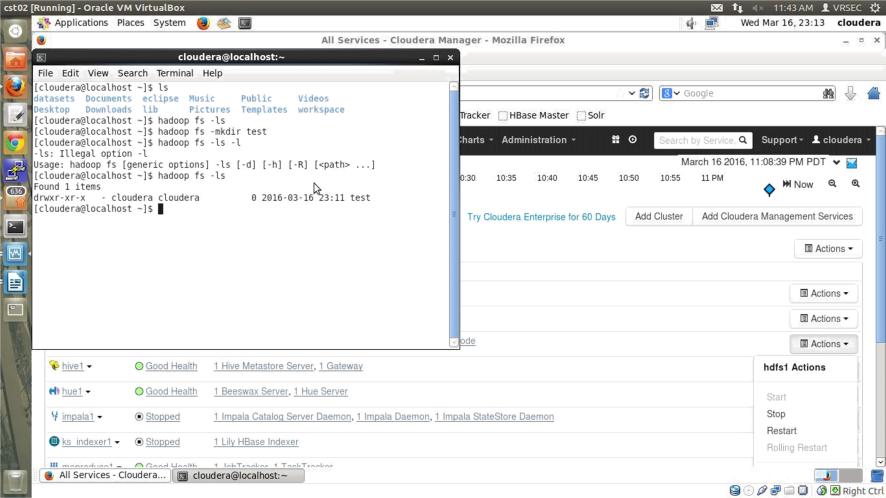
**Step4:** Services will be displayed.

**Step5:** check the health status of ‘hdfs’ and ‘zookeeper’ and Click on action and start the services



**Step6:** Click on Applications->systemtools->Terminal



**Step7:** Commands performed in linux terminal

**Result:** Installation of cloudera has been successfully described here.

**Task 2:**

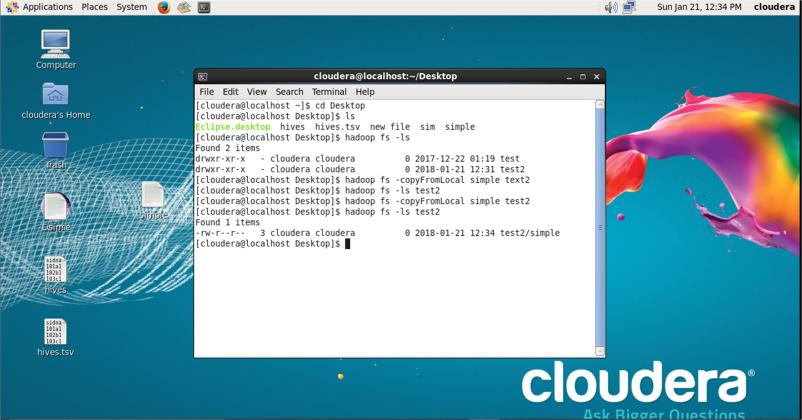
**Aim: To perform HDFS operations using commands**



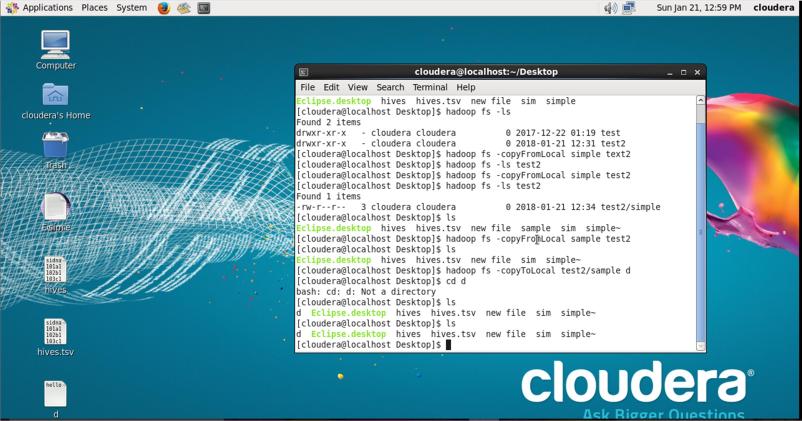
Steps:

1. To copy files from local file system to Hadoop filesystem.

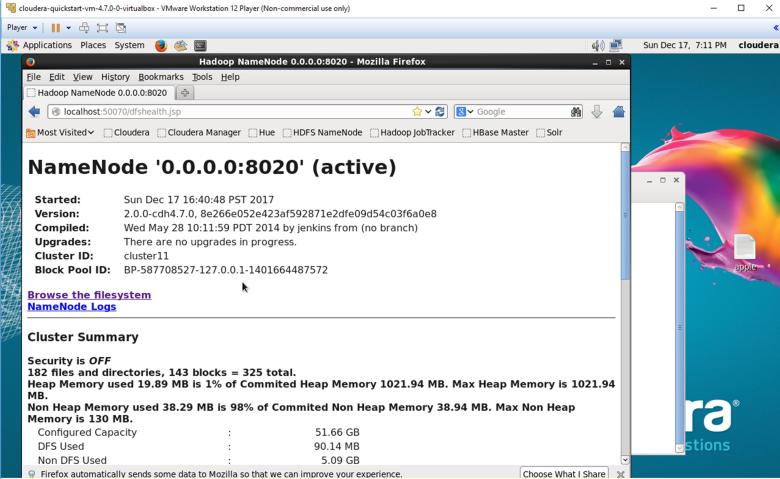
* Create file on local system(linux). (Ex.Desktop/sample.txt)
* Open terminal and type(Hadoop fs –copyFromLocal***Desktop/sample.txttest***)
* Type (Hadoop fs –lstest)



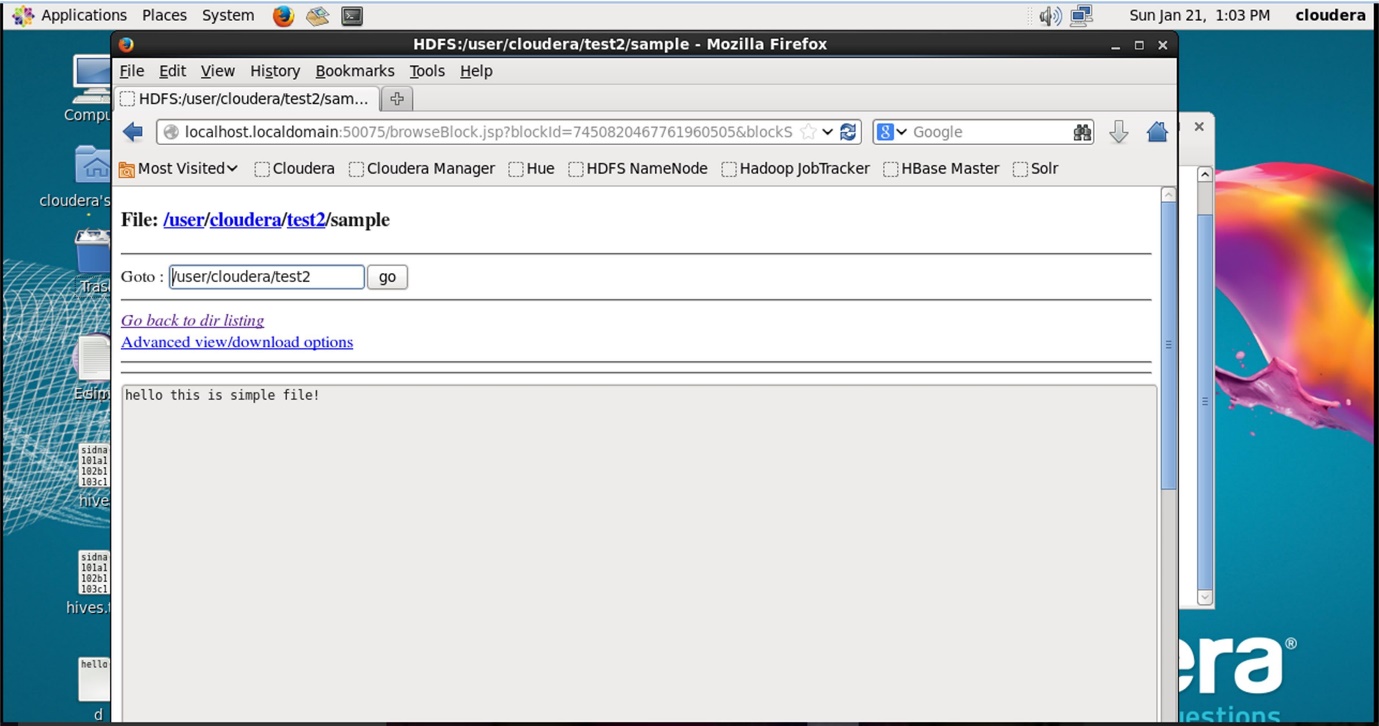
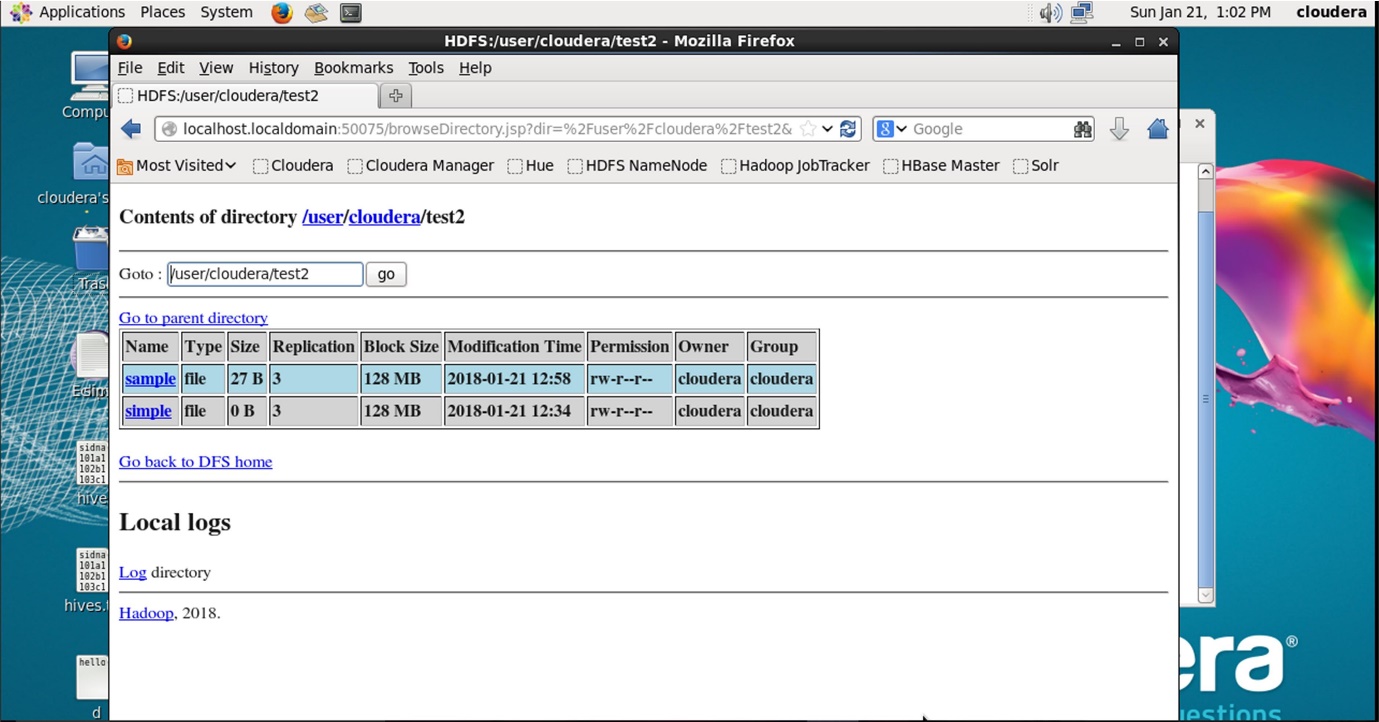
1. Hadoop commands to copy to Local fromhdfs
   * In the terminal type as following hadoop fs - copyToLocaldirectory\_name/existing\_file\_namenew\_file\_name
   * Now check the files in desktop usingls
   * The commands are as shownbelow



1. To see the files in the hadoop file system open the mozila fire fox and open Hdfsname node as shownbelow



1. Click on browse file system and see theFiles

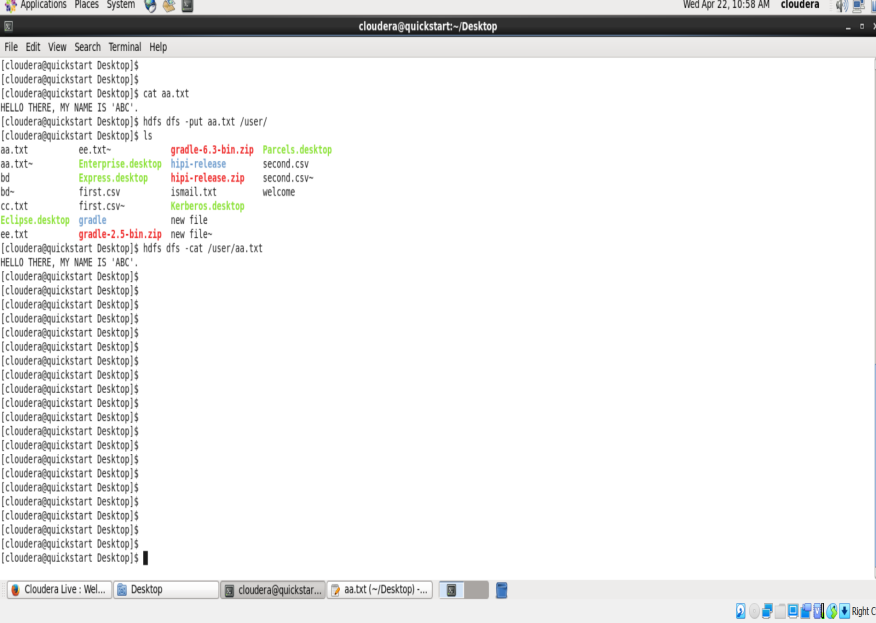


**Result:** The Hadoop operations using commands are performed.

**TASK – 3**

**Aim: Implementation of Cat command and file write operation on HDFS.**

**CAT COMMAND IMPLEMENTATION:**



**JAVA PROGRAM TO WRITE A FILE IN HDFS:**

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FSDataInputStream;

import org.apache.hadoop.fs.FSDataOutputStream;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

public class HDFSFileWrite {

    public static void main(String[] args) {

        Configuration conf = new Configuration();

        try {

            FileSystem fs = FileSystem.get(conf);

            // Hadoop DFS Path - Input & Output file

            Path inFile = new Path(args[0]);

            Path outFile = new Path(args[1]);

            // Verification

            if (!fs.exists(inFile)) {

                System.out.println("Input file not found");

                throw new IOException("Input file not found");

            }

            if (fs.exists(outFile)) {

                System.out.println("Output file already exists");

                throw new IOException("Output file already exists");

            }

            // open and read from file

            FSDataInputStream in = fs.open(inFile);

            // Create file to write

            FSDataOutputStream out = fs.create(outFile);

            byte buffer[] = new byte[256];

            try {

                int bytesRead = 0;

                while ((bytesRead = in.read(buffer)) > 0) {

                    out.write(buffer, 0, bytesRead);

                  }

            } catch (IOException e) {

                System.out.println("Error while copying file");

            } finally {

                in.close();

                out.close();

            }

        } catch (IOException e) {

            // TODO Auto-generated catch block

            e.printStackTrace();

        }

    }

}

**Executing program in Hadoop Environment**

To execute above Java program in Hadoop environment, you will need to add the directory containing the .class file for the Java program in Hadoop’s classpath.

* export HADOOP\_CLASSPATH='/huser/eclipse-workspace/knpcode/bin'

I have my HDFSFileWrite.class file in location **/huser/eclipse-workspace/knpcode/bin** so I have exported that path.

Then you can run the program by providing the path of the input file from which data is read and the path of the output file to which content is written.

* hadoop org.knpcode.HDFSFileWrite/user/input/test/aa.txt/user/input/test/write.txt

By using the ls HDFs command you can verify that the file is created or not.

* hdfs dfs-ls/user/input/test/
* -rw-r--r--1knpcode supergroup102018-01-1814:55/user/input/test/write.txt

**Writing HDFS file using IOUtils class**

Hadoop framework provides **IOUtils** class that has many convenient methods related to I/O. You can use that to copy bytes from the input stream to output stream.

**Java program to write HDFS file**

|  |
| --- |
| import java.io.IOException;    import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.FSDataInputStream;  import org.apache.hadoop.fs.FSDataOutputStream;  import org.apache.hadoop.fs.FileSystem;  import org.apache.hadoop.fs.Path;  import org.apache.hadoop.io.IOUtils;    publicclassHDFSFileWrite{        publicstaticvoidmain(String[]args){          Configuration conf=newConfiguration();          FSDataInputStream in=null;          FSDataOutputStream out=null;          try{              FileSystem fs=FileSystem.get(conf);              // Hadoop DFS Path - Input & Output file              Path inFile=newPath(args[0]);              Path outFile=newPath(args[1]);              // Verification              if(!fs.exists(inFile)){                  System.out.println("Input file not found");                  thrownewIOException("Input file not found");              }              if(fs.exists(outFile)){                  System.out.println("Output file already exists");                  thrownewIOException("Output file already exists");              }              try{                  // open and read from file                  in=fs.open(inFile);                  // Create file to write                   out=fs.create(outFile);                   IOUtils.copyBytes(in,out,512,false);                }finally{                  IOUtils.closeStream(in);                  IOUtils.closeStream(out);              }            }catch(IOExceptione){              // TODO Auto-generated catch block              e.printStackTrace();          }        }    } |

That’s all for the topic **Java Program to Write a File in HDFS**. If something is missing or you have something to share about the topic please write a comment

**Result:** The Implementation of Cat command and file write operations are proformed.

**TASK – 4**

**Aim: Execution of mapreduce programs like wordcount, tera data generator and tera data sort using Hadoop Examples jar**

**MAINCLASS:**

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.io.Text;

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

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

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

import org.apache.hadoop.util.ToolRunner;

import org.apache.hadoop.util.Tool;

public class WordCountJob implements Tool

{

private Configuration conf;

@Override

public Configuration getConf()

{

return conf;

}@Override

public void setConf(Configuration conf)

{

this.conf=conf;

}@Override

public int run(String []args)throws Exceptio{

Job wordcountjob=new Job(getConf());

wordcountjob.setJobName("mat word count");

wordcountjob.setJarByClass(this.getClass());

wordcountjob.setMapperClass(WordCountMapper.class);

wordcountjob.setReducerClass(WordCountReducer.class);

wordcountjob.setMapOutputKeyClass(Text.class);

wordcountjob.setMapOutputValueClass(LongWritable.class);

wordcountjob.setOutputKeyClass(Text.class);

wordcountjob.setOutputValueClass(LongWritable.class);

FileInputFormat.setInputPaths(wordcountjob,new Path(args[0]));

FileOutputFormat.setOutputPath(wordcountjob,new Path(args[1]));

wordcountjob.setNumReduceTasks(2);

return wordcountjob.waitForCompletion(true)==true? 0:1;

}

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

{

ToolRunner.run(new Configuration(),new WordCountJob(),args);

}

}

**MAPPER-CLASS:**

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

//import org.apache.hadoop.mapreduce.Counter;

public class WordCountMapper extends

Mapper<LongWritable, Text, Text, LongWritable> {

//private Text temp = new Text();

private final static LongWritable one = new LongWritable(1);

@Override

protected void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

String line = value.toString();

String[] words = line.split(" ");

for (int i = 0; i < words.length; i++) {

context.write(new Text(words[i]), one);

}

/\*StringTokenizer strtock = new StringTokenizer(str);

while (strtock.hasMoreTokens()) {

temp.set(strtock.nextToken());

context.write(temp, one);

}\*/

}

}

**REDUCER-CLASS:**

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.io.Text;

public class WordCountReducer extends Reducer<Text,LongWritable,Text,LongWritable>

{

@Override

protected void reduce(Text key,Iterable<LongWritable> value,Context context)throws IOException,InterruptedException

{

long sum=0;

while(value.iterator().hasNext())

{

sum+=value.iterator().next().get();

}

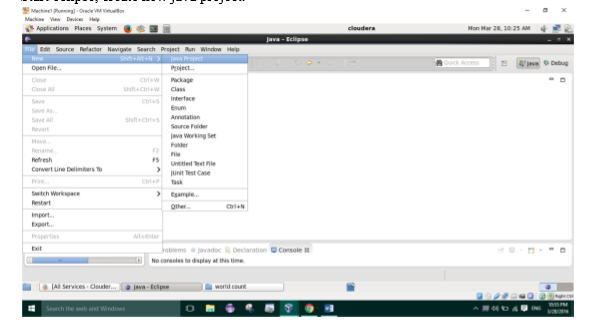
context.write(key,new LongWritable(sum));

}

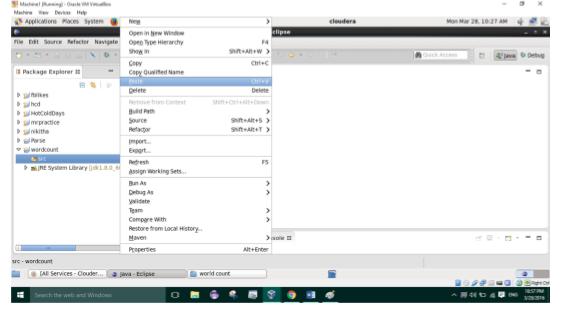
}

**STEP1:**

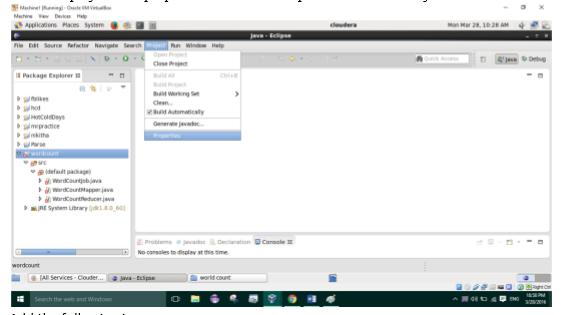
Start eclipse, create new java project



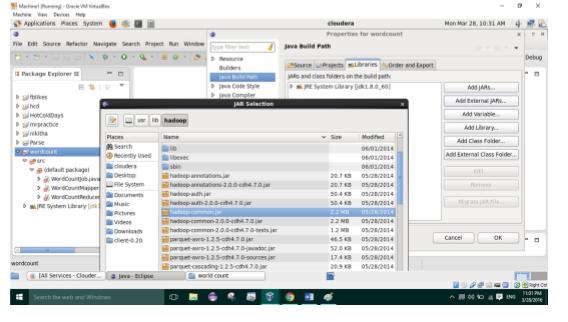
* Open ‘src’ and paste the mapper,reducer and job java files in default package.

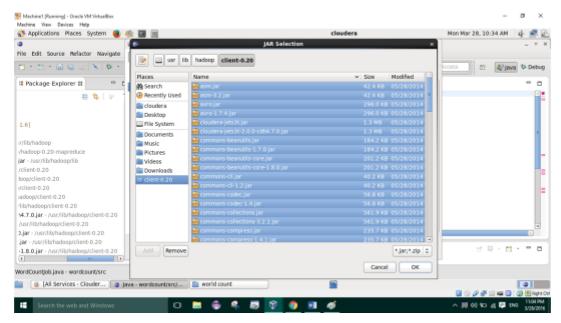


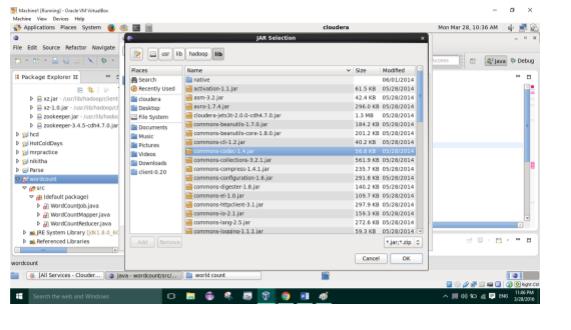
* Click on ‘project’ -> ‘properties’ ->Java build path ->add external jars



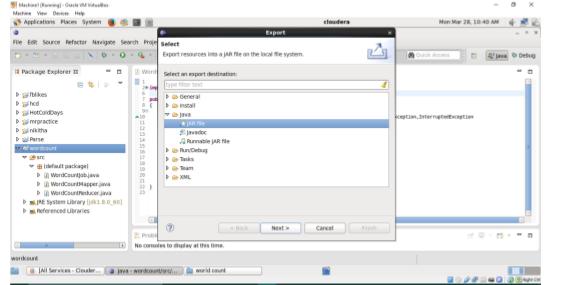
* Add the following jars
* Usr/lib/Hadoop/Hadoop-common.jar
* Usr/lib/Hadoop/ client-0.20/ (all jars)
* Usr/lib/Hadoop/lib/commons-code-1.4.jar

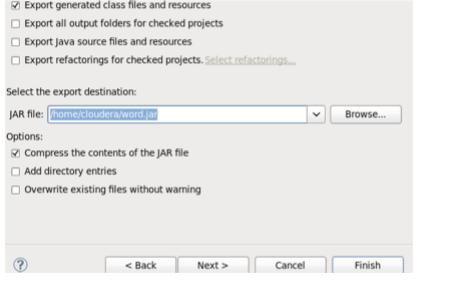




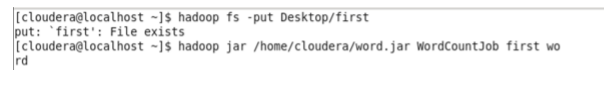


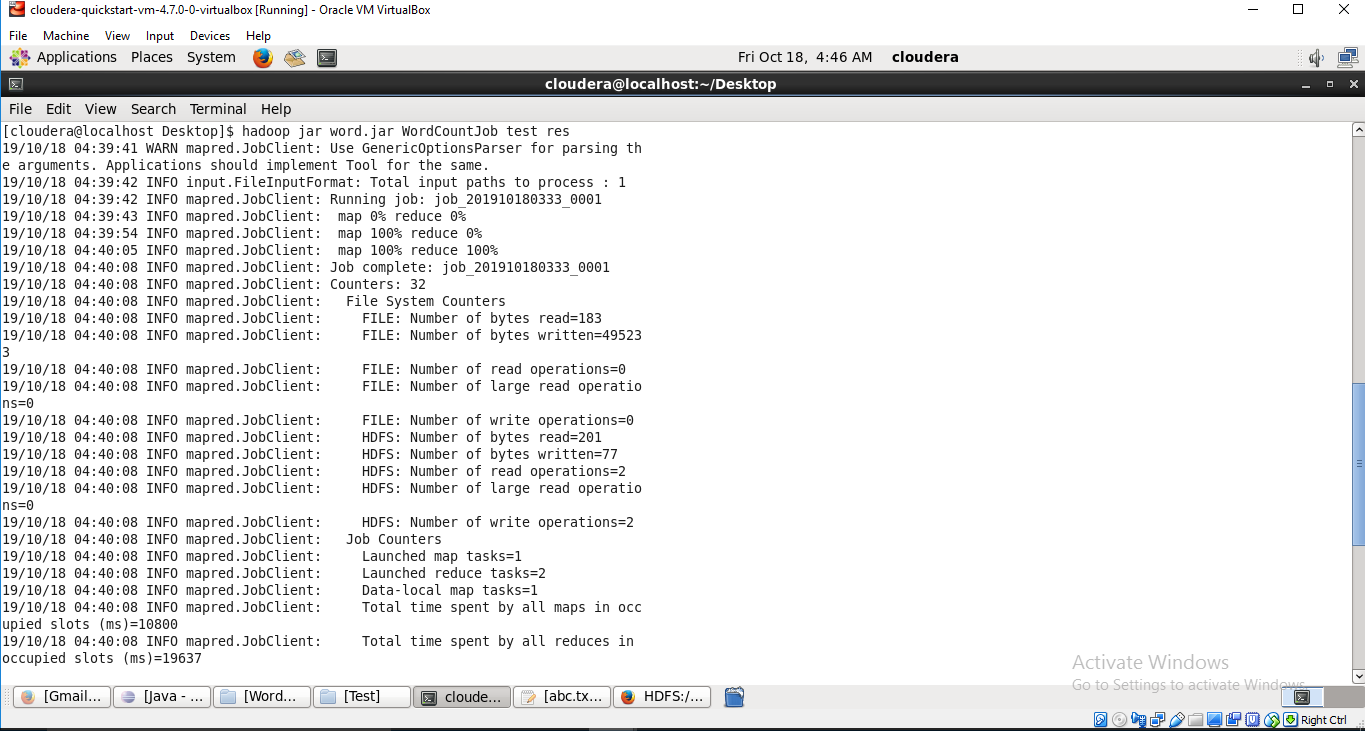
* Resave all .java filesClick on file and export as jar file, give name and path where jar file can be stored(Ex: Desktop)



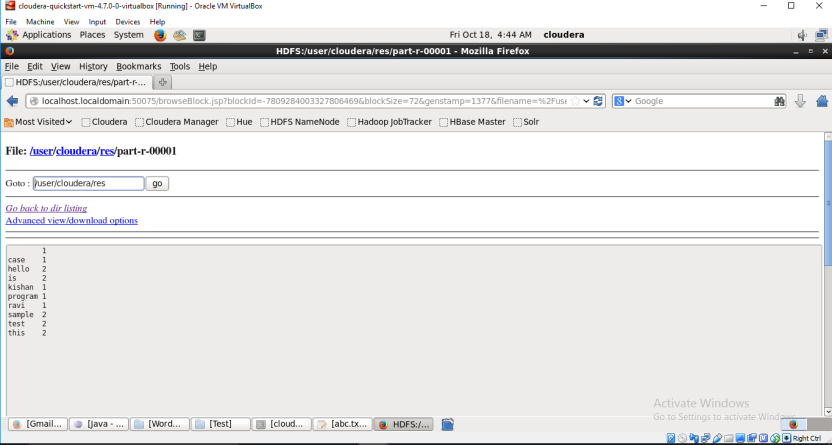


* Store test file in hdfs and run wordcount.jar on that file by using following command o Hadoop jar wordcount.jar [mainclassfilename] [input hdfs path] [output hdfs folder]





**Output:** The execution of mapreduce programs like wordcount is completed.

****

**Result:** The word count has been successfully found using map reduce.

**TASK – 5**

**Aim: Implementation of Word Count job in Hadoop using python.**

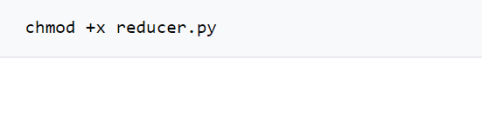
**MAPPER:**



**REDUCER:**

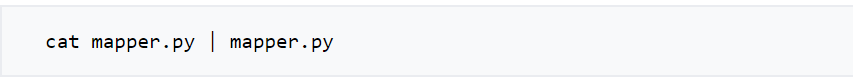


Make sure the file is executable



## Testing

* Make sure two programs work. Here's a simple series of test you can run:



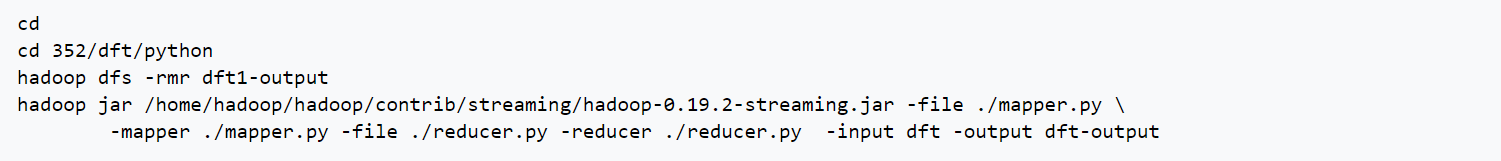
This will make mapper.py output all the words that make up its code.



This will generate the (unsorted) frequencies of all the unique words (punctuated or not) in mapper.py.

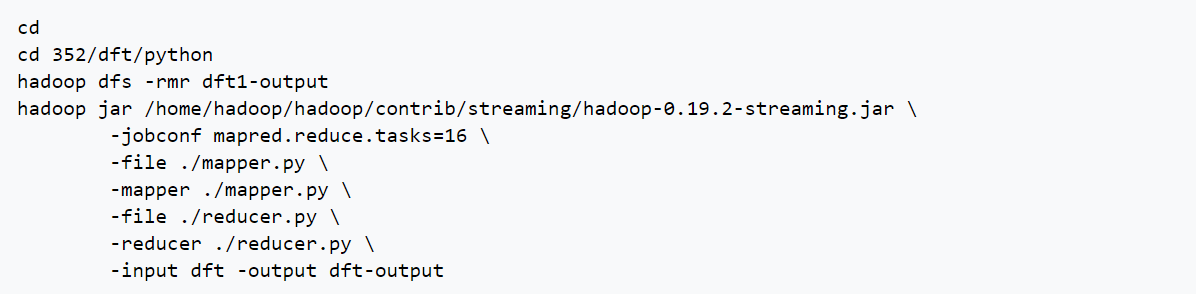
## Running on the Hadoop Cluster

* Let's run the Python code on the sample1.txt file.
* We'll assume that the Python code is stored in ~hadoop/352/dft/python
* We'll assume that the streaming java library is in ~hadoop/contrib/streaming/streaming-0.19.2-streaming.jar
* We'll also assume that ulysses.txt is in dft and that we want the output in dft-output:



Changing the number of Reducers

* To change the number of reducers, simply add this switch **-jobconf mapred.reduce.tasks=16** to the command line:



**Result**: Implementation of Word Count job in Hadoop using python is performed.

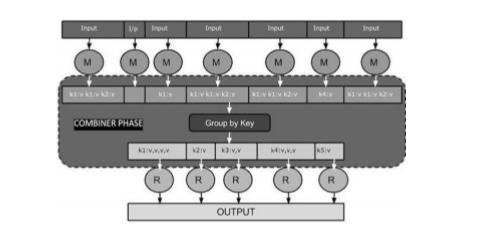
**TASK – 6**

**Implementation of Word Count using MapReduce with combiner.**

**Combiner**

* A Combiner, also known as a semi-reducer, is an optional class that operates by accepting the inputs from the Map class and thereafter passing the output key-value pairs to the Reducer class.
* The main function of a Combiner is to summarize the map output records with the same key. The output (key-value collection) of the combiner will be sent over the network to the actual Reducer task as input.
* The Combiner class is used in between the Map class and the Reduce class to reduce the volume of data transfer between Map and Reduce. Usually, the output of the map task is large and the data transferred to the reduce task is high.

The following MapReduce task diagram shows the COMBINER PHASE.



**Combiner working**

* A combiner does not have a predefined interface and it must implement the Reducer interface’s reduce() method.
* A combiner operates on each map output key. It must have the same output key-value types as the Reducer class.
* A combiner can produce summary information from a large dataset because it replaces the original Map output.
* Although, Combiner is optional yet it helps segregating data into multiple groups for Reduce phase, which makes it easier to process.

**MapReduce Combiner Implementation**

Let us assume we have the following input text file named input.txt for MapReduce.

What do you mean by Object

What do you know about Java

What is Java Virtual Machine

How Java enabled High Performance

The important phases of the MapReduce program with Combiner are discussed below.

**Record Reader**

This is the first phase of MapReduce where the Record Reader reads every line from the input text file as text and yields output as key-value pairs.

**Input** − Line by line text from the input file.

**Output** − Forms the key-value pairs. The following is the set of expected key-value pairs.

<1, What do you mean by Object>

<2, What do you know about Java>

<3, What is Java Virtual Machine>

<4, How Java enabled High Performance>

**Map Phase**

The Map phase takes input from the Record Reader, processes it, and produces the output as another set of key-value pairs.

**Input** − The following key-value pair is the input taken from the Record Reader.

The Map phase reads each key-value pair, divides each word from the value using String Tokenizer, treats each word as key and the count of that word as value. The following code snippet shows the Mapper class and the map function.

**Output** − The expected output is as follows −

<What,1><do,1><you,1><mean,1><by,1><Object,1>

<What,1><do,1><you,1><know,1><about,1><Java,1>

<What,1><is,1><Java,1><Virtual,1><Machine,1>

<How,1><Java,1><enabled,1><High,1><Performance,1>

**Combiner Phase**

The Combiner phase takes each key-value pair from the Map phase, processes it, and produces the output as key-value collection pairs.

**Input** − The following key-value pair is the input taken from the Map phase.

The Combiner phase reads each key-value pair, combines the common words as key and values as collection.

**Output** − The expected output is as follows

<What,1,1,1><do,1,1><you,1,1><mean,1><by,1><Object,1>

<know,1><about,1><Java,1,1,1>

<is,1><Virtual,1><Machine,1>

<How,1><enabled,1><High,1><Performance,1>

**Reducer Phase**

The Reducer phase takes each key-value collection pair from the Combiner phase, processes it, and passes the output as key-value pairs.

**Input** − The input is taken from the Combiner phase.

The Reducer phase reads each key-value pair. Following is the code snippet for the Combiner.

**Output** − The expected output from the Reducer phase is as follows −

<What,3><do,2><you,2><mean,1><by,1><Object,1>

<know,1><about,1><Java,3>

<is,1><Virtual,1><Machine,1>

<How,1><enabled,1><High,1><Performance,1>

**Result**: The Mapper,Reducer,Combiner concepts are explained with an example successfully.

**TASK – 7**

**Aim: Implementation of MapReduce program that mines weather data.**

**STEPS:**

**MAINCLASS:**

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapred.\*;

import org.apache.hadoop.util.\*;

public class HighestDriver extends Configured implements Tool{

public int run(String[] args) throws Exception

{

JobConf conf = new JobConf(getConf(), HighestDriver.class);

conf.setJobName("HighestDriver");

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

conf.setMapperClass(HighestMapper.class);

conf.setReducerClass(HighestReducer.class);

Path inp = new Path(args[0]);

Path out = new Path(args[1]);

FileInputFormat.addInputPath(conf, inp);

FileOutputFormat.setOutputPath(conf, out);

JobClient.runJob(conf);

return 0;

}

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

{

int res = ToolRunner.run(new Configuration(), new HighestDriver(),args);

System.exit(res);

}

}

**MAPPER-CLASS:**

import java.io.IOException;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapred.\*;

public class HighestMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>

{

public static final int MISSING = 9999;

public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException

{

String line = value.toString();

String year = line.substring(15,19);

int temperature;

if (line.charAt(87)=='+')

temperature = Integer.parseInt(line.substring(88, 92));

else

temperature = Integer.parseInt(line.substring(87, 92));

String quality = line.substring(92, 93);

if(temperature != MISSING && quality.matches("[01459]"))

output.collect(new Text(year),new IntWritable(temperature));

}

}

**REDUCER-CLASS:**

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapred.\*;

public class HighestReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable>

{

public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException

{

int max\_temp = 0;

;

while (values.hasNext())

{

int current=values.next().get();

if ( max\_temp < current)

max\_temp = current;

}

output.collect(key, new IntWritable(max\_temp/10));

}

}

**STEP1:**

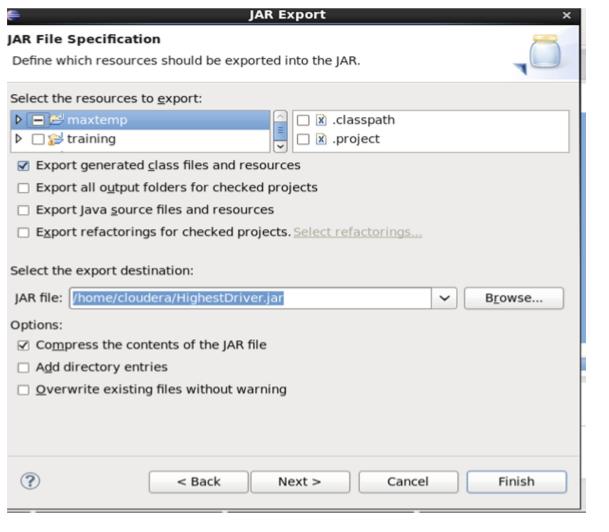
NEW->SELECT NEW JAVAPROJECT->NAME AS HIGHESTDRIVER

SRC->RIGHTCLICK->BUILDPATH->CONFIGPATH->LIBRARIES>ADDEXTERNALJARS

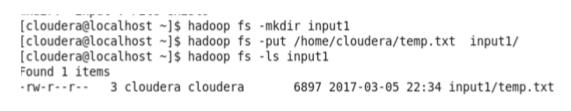
SRC->RIGHTCLICK->NEW->CLASSFILE

CREATE 3 CLASS FILES FOR MAIN,MAPPER,REDUCER CLASSES.

SRC->RIGHTCLICK->EXPORT->JAVA->JARFILE->GIVE PATHNAME->FINISH

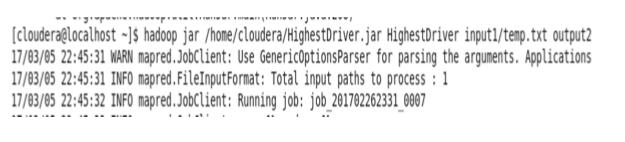


**STEP2:** OPEN TERMINAL

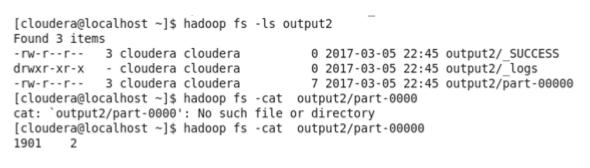


Temp.txt is the temperature datafile

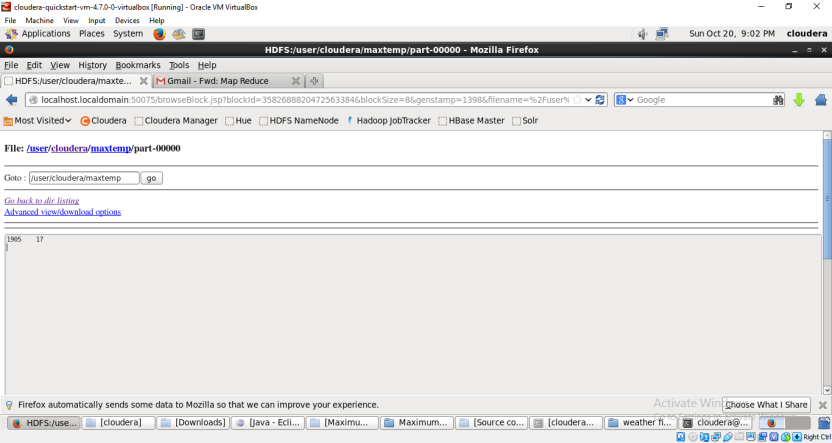
**STEP3:**



CREATING PATHS AND JAR FILES IN TERMINAL



**OUTPUT:**

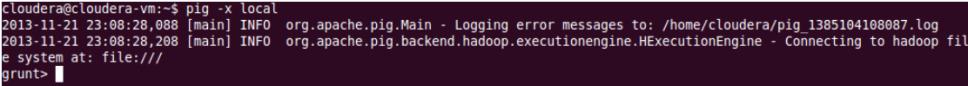


**Result**: The Max Temperature in the respective years has been found successfully.

**TASK – 8**

**Aim: Implementation of data processing operators using Pig Latin.**

* In local mode loading the grunt shell.

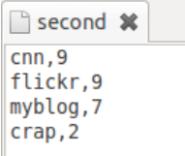


* Let’s create two files first and second to run the commands:

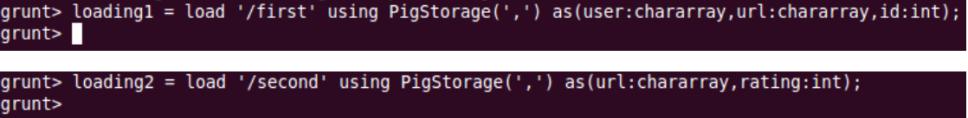
 The first file contain three fields: user, url & id.

The second file contain two fields: url & rating. These two files are CSV files.





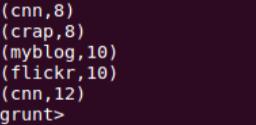
* The Load operator loads data from file ‘first’ to form relation ‘loading1’. The field names are user, url, id.



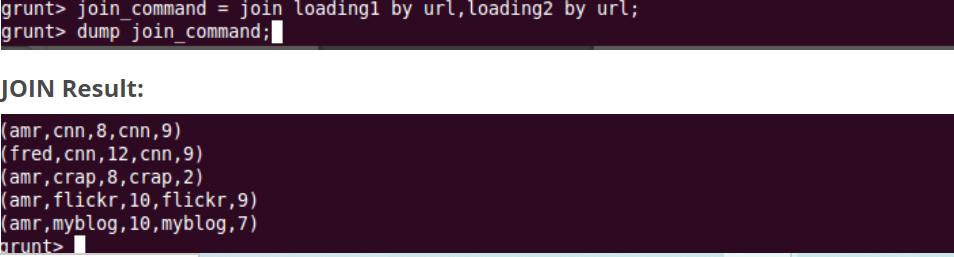
* Foreach operator is used to add or remove fields from a relation.



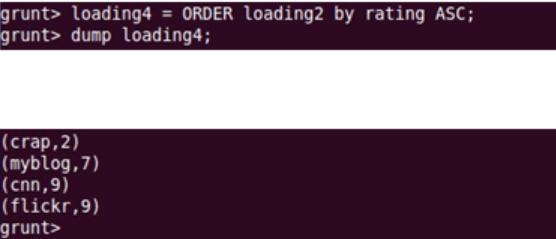
Foreach results:



* We are filtering the record from ‘loading1’ when the condition ‘id’ is greater than 8.



* We are sorting data in loading2 in ascending order on ratings field using ORDER\_BY.



**Result:** Implementation of data processing operators using Pig Latin is performed.

**TASK – 9**

**Aim: Implementation of data analytics using pig.**

* **Load Grusnt shell.**

pig -x local

* **Create a pig relation using HDFS data stored stocks dir.**

grunt> STOCKS = LOAD '/user/cloudera/stocks'

USING PigStorage(',') AS

(date: chararray,

open: double,

high: double,

low: double,

close: double,

volume: double,

adjClose: double,

symbol: chararray);

* **View relation description**

grunt> DESCRIBE STOCKS;

* **View fields, data type of fields and a sample row from a relation.**

grunt> ILLUSTRATE STOCKS;

* **Create a new relation with 5 columns from STOCKS relation.**

grunt> STOCKS\_TOP5 = LIMIT STOCKS 5;

grunt> DUMP STOCKS\_TOP5;

* **Project symbol and volume columns from STOCKS relation. Cast the volume column to long data type.**

grunt> SYMBOL\_VOLUME = FOREACH STOCKS GENERATE symbol, (long) volume as volume;

* **Store the SYMBOL\_VOLUME relation into HDFS in csv format.**

grunt> STORE SYMBOL\_VOLUME INTO '/user/cloudera/stocks.lean' USING PigStorage(',');

* **Create a hcatalog table for SYMBOL\_VOLUME relation.**

$ hcat -e "create table stocks\_lean(symbol string, volume bigint) row format delimited fields terminated by ':' stored as textfile"

$ hcat -e "show tables"

$ hcat -e "describe stocks\_lean"

* **Count the total number of symbol.**

grunt> SYMBOLS\_COUNT = FOREACH (GROUP SYMBOLS ALL) GENERATE COUNT(SYMBOLS) PARALLEL 3;

* **Find date range from STOCKS relation**

grunt> GROUPED = GROUP STOCKS ALL;

grunt> DATE\_AGGR = FOREACH GROUPED GENERATE MIN(STOCKS.date) AS min, MAX(STOCKS.date) AS max;

grunt> DUMP DATE\_AGGR;

* **Find average volume of stocks traded in 2016**

grunt> STOCKS2016 = FILTER STOCKS BY STARTSWITH(date, '2016');

grunt> GROUPED = GROUP STOCKS BY symbol;

grunt> VOL\_AGGR = FOREACH GROUPED GENERATE group AS symbol, AVG(STOCKS.volume) AS avg;

grunt> VOL\_AGGR\_ORDERED = ORDER VOL\_AGGR BY avg DESC;

grunt> DUMP VOL\_AGGR\_ORDERED;

**Result:** Implementation of data analytics using pig

**TASK - 10**

**Aim: Creating Hive tables and Loading data using Hive Query language.**

### Procedure:

**Step-1:**

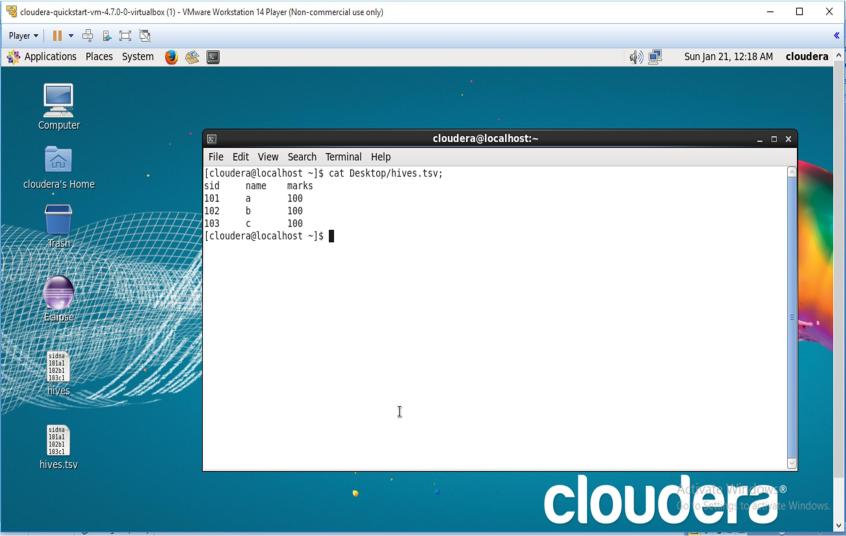
Open terminal and type “hive” it displays hive terminal.

### Step-2:

Create the file in desktop using command in terminal

Cat Desktop/file\_name

Now enter the data tab delimited as shown below

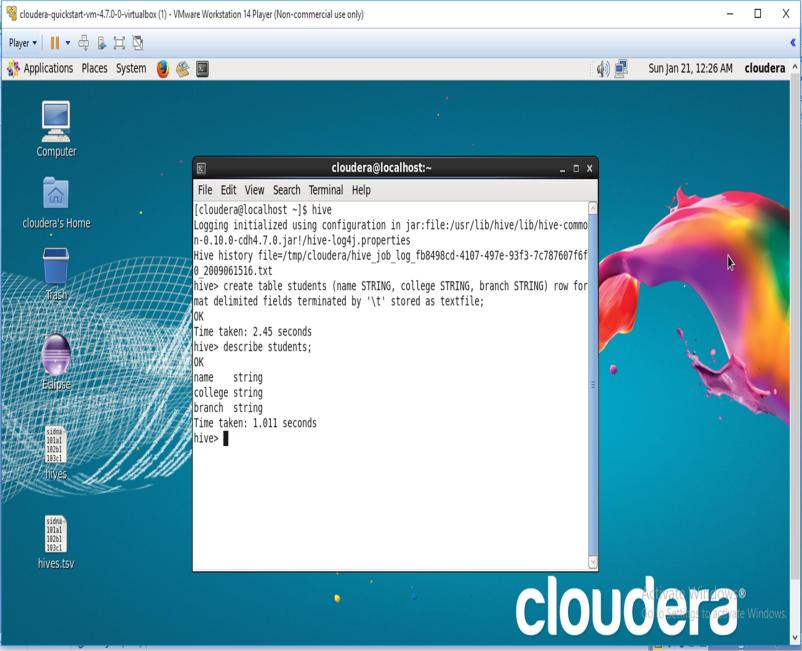


### Step-3:

Enter create command in hive it is as shown below

Create table student(name STRING, college STRING, branch STRING) row format delimited by ‘\t’ stored to text file.

After that describe Student;



**Result:**The Creation of tables in hiveql language has been successfully described

**TASK – 11**

**Aim : Implement partitioning of data in Hive Warehouse using Hive Query Language.**

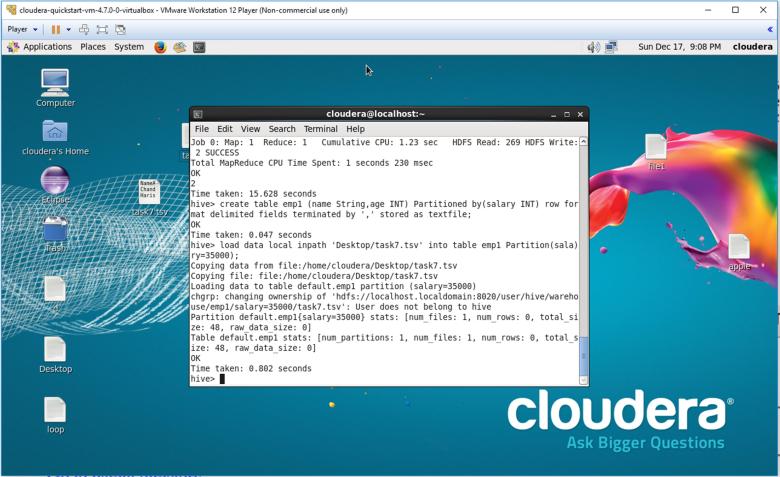
Steps:

Create a tsv file which contains many number of column names on desktop of cloudera.

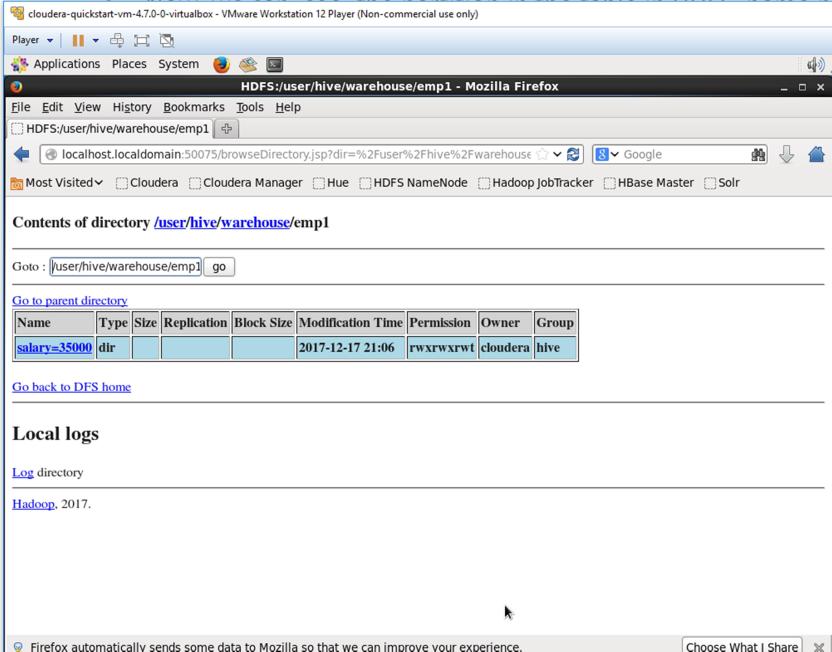
Load the data into the hive using query.

open terminal and type hive

Now create a table using partitioned query in hiveQL.



Now we can see the partition in the table in HDFS name node



**Result:** The tables are partitioned successfully.

**TASK – 12**

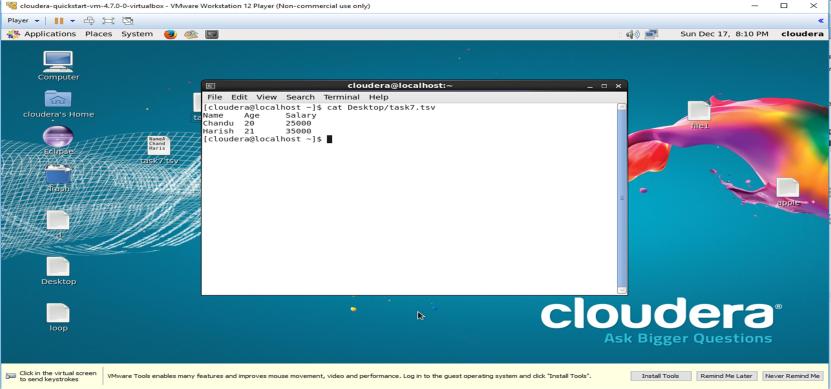
**Aim: Loading data into HIVE warehouse. Apply aggregate operations on data.**

Steps:

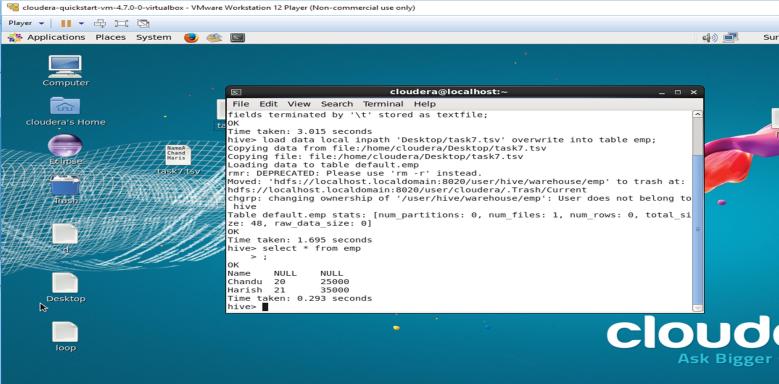
Create tsv file with numeric data.

Open terminal and enter hive, you are entered into hive shell.

Create table using “create command” of hiveQL.

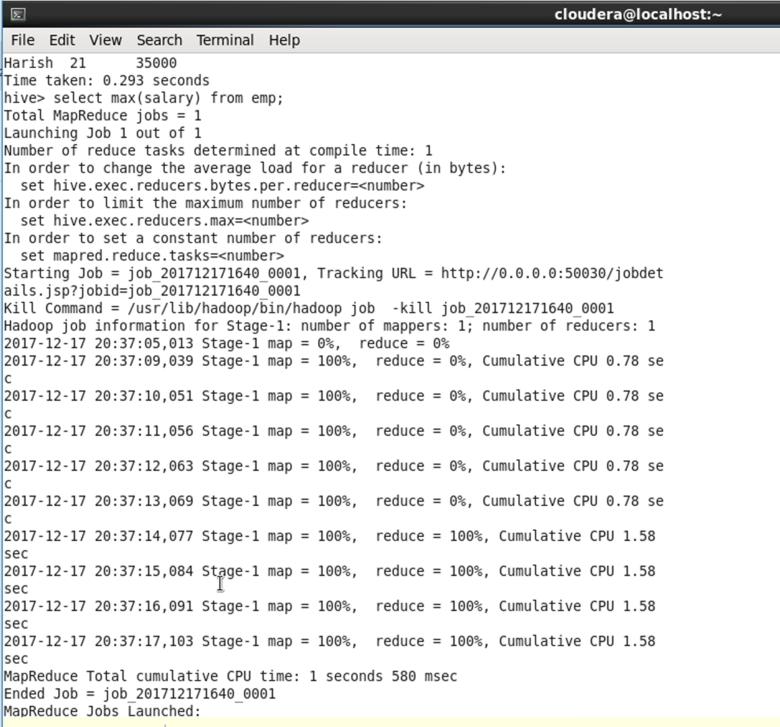


Type load command of hiveQL to load data from client machine to hive warehouse.

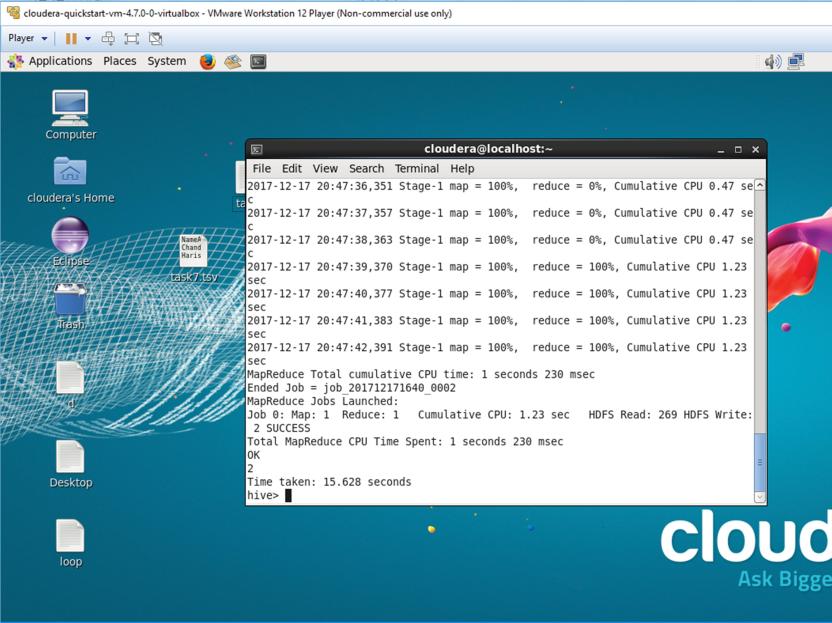


Apply aggregate operations on data like max, sum, and count.

### Max



**Count**



**Result:** Loading of data into hive tables and performing operations on it has been successfully explained.

# Internal and External Question Papers with CO-PO-PSO

# Mapping Internal Question List

|  |  |  |  |
| --- | --- | --- | --- |
| **SNo** | **Experiment** | **CO Mapping** | **PO Mapping** |
| 1 | A) Implement MapReduce program to find word count with combiner without combiner. Note down the time difference.  b) Write HiveQL statement to extract data from xml file and load into hive table  xml file have student no, name, department.  Find department wise student count. | **CO1,CO2** | **PO1,PO2,PO4,PO5,PO9** |
| 2 | A) Implement MapReduce program to find the years whose average sales is greater than 3000.  input file format has year, sales of all months and average sales  year janfeb march april may junejuly august septemberoctobernovemberdecember average  b) Write a subquery to count the occurrence of similar words in the file using Hive Query Language. | **CO1,CO2** | **PO1,PO2,PO4,PO5,PO9** |
| 3 | 1. A) Find and max salary of each department and their employee details using MapReduce   Empno EmpName Dept Salary  b) Create the customer table with the following schema using HQL  custid, customername, lifetime value  1001 , Jack, 25000  Create a partition table if life time value is 12000.  Create a partition table for all life time values. | **CO1,CO2** | **PO1,PO2,PO4,PO5,PO9** |
| 4 | A) Write MapReduce program to find customer wise total sale amount  Custid name amount  b) Create the following tables  Orders: custid, itemid, itemname, orderdate, deliverydate  Customers: custid, custname, address, city, state, country.  Write a HiveQL to find the number of items bought by each customer. | **CO1,CO2** | **PO1,PO2,PO4,PO5,PO9** |
| 5 | A) Implement MapReduce program to find Dept wise salary. **Data Set:** Name Age Dept Gender Salary A 23 IT Male 35000 B 35 Finance Female 50000 C 29 IT Male 40000  B) Create the following tables  Orders: custid,itemid, itemname,orderdate,deliverydate  Customers: custid,custname,address, city, state, country.  Create joined table with the following attributes  Custid, name, itemid, name,orderdate, deliverydate, address | **CO1,CO2** | **PO1,PO2,PO4,PO5,PO9** |
| 6 | A) Implement MapReduce based solution to find the particular employee details.  Empid, name, age, address, phone,  1001, john, 45  1002, jack, 39  1003,Bob,33  Example: find jack details.  b) Create the student table with the following schema using HQL.  stdno name deptgpa  find the details of students who got maximum, minim gpa.  Load the data from a text file.  Create buckets based on dept.  Find department wise count from bucketed table.  Identify the time difference between normal table and bucketed table and give reasons. | **CO1,CO2** | **PO1,PO2,PO4,PO5,PO9** |
| 7 | A) Implement mapreduce program to find the maximum temperature of each year from weather data.  assume weather data has year and temperature  b)Create customer table and load data using HIVE QL.  Customers: custid,custname,address, city, state, country.  create view for the above table with the following attributes  custid, city, country | **CO1,CO2** | **PO1,PO2,PO4,PO5,PO9** |
| 8 | 1. A) Implement MapReduce program to find count of middle age, senior and youth in the organization. **Data Set:** Name Age Dept Gender Salary A 23 IT Male 35000 B 35 Finance Female 50000 C 29 IT Male 40000   B) Create the following tables using HIVE QL.  Orders: stdid,itemid, itemname,orderdate,deliverydate  Customers: custid,custname,address, city, state, country.  Create joined table with the following attributes  Custid, name, itemid, name,orderdate, deliverydate, address | **CO1,CO2** | **PO1,PO2,PO4,PO5,PO9** |
| 9 | a) Implement mapreduce program to find citywise student count.  dataset  student roll no Name dept  S001 Ramu IT  b) create table and load data using Hive, display the students of particular city.  stdrollno name address city | **CO1,CO2** | **PO1,PO2,PO4,PO5,PO9** |
| 10 | A) Implement MapReduce program to find Dept wise employee count.  EmpnoEmpNameDept Salary  b) Create the following tables using HIVE QL  Orders: custid,itemid, itemname,orderdate,deliverydate  Customers: custid,custname,address, city, state, country.  Write a HiveQL to find the number of items bought by each customer. | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| 11 | A) Implement MapReduce program to find word count with combiner without combiner. Note down the time difference.  b) load the student data using Piglatin  student no, name, department.  Find department wise student count. | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| 12 | A) Implement MapReduce program to find the years whose average sales is greater than 3000.  input file format has year, sales of all months and average sales  year janfeb march april may junejuly august septemberoctobernovemberdecember average  b) Load the text file and find top 5 words using Pig Latin. | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| 13 | A) Find and max salary of each department and their employee details using MapReduce  Empno EmpName Dept Salary  b) Load customer data using PigLatin.  custid, customername, lifetime value  1001 , Jack, 25000  display customers whose lifetime value is greater than 100000 | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| 14 | A) Write MapReduce program to find customer wise total sale amount  Custid name amount  b) Create the following tables  Orders: custid, itemid, itemname, orderdate, deliverydate  Customers: custid, custname, address, city, state, country.  Write PigLatin to find the number of items ordered in Oct 2020. | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| 15 | 1. A) Implement MapReduce program to Dept wise salary. **Data Set:** Name Age Dept Gender Salary A 23 IT Male 35000 B) Load the data into student relation   rollno,name,gpa  Load the data into department relation  rollno,deptno,deptname  Join the above two relatlions using Pig Latin | **CO1,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| 16 | 1. A) Implement MapReduce based solution to find the particular employee details.   Empid, name, age, address, phone,  1001, john, 45  1002, jack, 39  1003,Bob,33  Example: find jack details.  b) Load the data into relations in Pig Latin .  stdno, name, dept, gpa  find the details of students who got maximum, minim gpa in department wise. | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| 17 | a) Implement mapreduce program to find the maximum temperature of each year from weather data.  assume weather data has year and temperature  b)Load the customers data and find country wise customer count using PigLatin  Customers: custid,custname,address, city, state, country. | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| 18 | A) Implement MapReduce based solution to find Itemwise total Sale amount.  Itemid description saleamount  b) Load the student data and display student data based on Gpa descending order using PigLatin  Rollno ,Name, Gpa | **CO1,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| 19 | A) Implement MapReduce program to find count of middle age, senior and youth in the organization. (<30 youth, 30-60 middle, >60 senior) **Data Set:** Name Age Dept Gender Salary A 23 IT Male 35000 B 35 Finance Female 50000 C 29 IT Male 40000  B) Load the above empolyee data into relation using PigLatin and display all senior employees. | **CO1,CO2** | **PO1,PO2,PO4,PO5,PO9** |
| 20 | A) Implement mapreduce program to find citywise student count.dataset  student roll no Name dept  S001 Ramu IT  b) Load student data using Pig Latin, display the students of vijayawada city.  stdrollno name address city | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| 21 | A) Implement MapReduce program to find Dept wise employee count.  EmpnoEmpNameDept Salary  b) Load the following data into relations and join using Pig Latin  Orders: custid,itemid, itemname,orderdate,deliverydate  Customers: custid,custname,address, city, state, country. | **CO1,CO3** | **PO1,PO2,PO4,PO5,PO9** |

**External Question List**

|  |  |  |  |
| --- | --- | --- | --- |
| **SNo** | **Experiment** | **CO Mapping** | **PO Mapping** |
| **1** | A) Implement MapReduce program to find word count with combiner without combiner. Note down the time difference.  b) Write HiveQL statement to extract data from xml file and load into hive table  xml file have student no, name, department.  Find department wise student count. | **CO1,CO2** | **PO1,PO2,PO4,PO5,PO9** |
| **2** | A) Implement MapReduce program to find the years whose average sales is greater than 3000.  input file format has year, sales of all months and average sales  year janfeb march april may junejuly august septemberoctobernovemberdecember average  b) Write a subquery to count the occurrence of similar words in the file using Hive Query Language. | **CO2, CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **3** | A) Find and max salary of each department and their employee details using MapReduce  Empno EmpName Dept Salary  b) Create the customer table with the following schema using HQL  custid, customername, lifetime value  1001 , Jack, 25000  Create a partition table if life time value is 12000.  Create a partition table for all life time values. | **CO2, CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **4** | Write MapReduce program to find customer wise total sale amount  Custid name amount  b) Create the following tables  Orders: custid, itemid, itemname, orderdate, deliverydate  Customers: custid, custname, address, city, state, country.  Write a HiveQL to find the number of items bought by each customer. | **CO2** | **PO1,PO2,PO4,PO5,PO9** |
| **5** | A) Implement MapReduceprogram to Dept wise salary. **Data Set:** Name Age Dept Gender Salary A 23 IT Male 35000 B 35 Finance Female 50000 C 29 IT Male 40000  B) Create the following tables  Orders: custid,itemid, itemname,orderdate,deliverydate  Customers: custid,custname,address, city, state, country.  Create joined table with the following attributes  Custid, name, itemid, name,orderdate, deliverydate, address | **CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **6** | A) Implement MapReduce based solution to find the particular employee details.  Empid, name, age, address, phone,  1001, john, 45  1002, jack, 39  1003,Bob,33  Example: find jack details.  b) Create the student table with the following schema using HQL.  stdno name deptgpa  find the details of students who got maximum, minim gpa.  Load the data from a text file.  Create buckets based on dept.  Find department wise count from bucketed table.  Identify the time difference between normal table and bucketed table and give reasons. | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **7** | a) Implement mapreduce program to find the maximum temperature of each year from weather data.  assume weather data has year and temperature  b)Create customer table and load data using HIVE QL.  Customers: custid,custname,address, city, state, country.  create view for the above table with the following attributes custid, city, country | **CO2**  **CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **8** | a) Implement MapReduce based solution to find Itemwise total Sale amount.  Itemid description saleamount  b) Create table with following schema  rollno int Name string Gpa float  Find maximum, min and average gpa  Load the data into RC table  Find the maximum, min and average gpa from RC table.  Identify the time difference and give reasons. | **CO1** | **PO1,PO2,PO4,PO5,PO9** |
| **9** | A) Implement MapReduce program to find count of middle age, senior and youth in the organization. **Data Set:** Name Age Dept Gender Salary A 23 IT Male 35000 B 35 Finance Female 50000 C 29 IT Male 40000  B) Create the following tables using HIVE QL.  Orders: stdid,itemid, itemname,orderdate,deliverydate  Customers: custid,custname,address, city, state, country.  Create joined table with the following attributes  Custid, name, itemid, name,orderdate,  deliverydate, address | **CO2, CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **10** | A) Implement mapreduce program to find citywise student count.  dataset  student roll no Name dept  S001 Ramu IT  b) create table and load data using Hive, display the students of particular city.  stdrollno name address city | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **11** | A) Implement MapReduce program to find Dept wise employee count.  EmpnoEmpNameDept Salary  b) Create the following tables using HIVE QL  Orders: custid,itemid, itemname,orderdate,deliverydate  Customers: custid,custname,address, city, state, country.  Write a HiveQL to find the number of items bought by each customer. | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **12** | a) Implement Word count using MapReduce.  b) Create Customer table with the following schema :custid,custname,address  Address is a structure : city, state, country.  CREATE TABLE STUDENT\_INFO  rollnoint,  name string,  friends ARRAY STRING,  marks map type having <subject,marks>  create student table Load the data and query the particular student details. | **CO1,CO2** | **PO1,PO2,PO4,PO5,PO9** |
| **13** | A) Implement MapReduce program to find word count with combiner without combiner. Note down the time difference.  b) load the student data using Piglatin  student no, name, department.  Find department wise student count. | **CO1,CO2** | **PO1,PO2,PO4,PO5,PO9** |
| **14** | A) Implement MapReduce program to find the years whose average sales is greater than 3000.  input file format has year, sales of all months and average sales  year janfeb march april may junejuly august septemberoctobernovemberdecember average  b) Load the text file and find top 5 words using Pig Latin. | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **15** | A) Find and max salary of each department and their employee details using MapReduce  Empno EmpName Dept Salary  b) Load customer data using PigLatin.  custid, customername, lifetime value  1001 , Jack, 25000  display customers whose lifetime value is greater than 100000 | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **16** | A) Write MapReduce program to find customer wise total sale amount  Custid name amount  b) Create the following tables  Orders: custid, itemid, itemname, orderdate, deliverydate  Customers: custid, custname, address, city, state, country.  Write PigLatin to find the number of items ordered in Oct 2020. | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **17** | A) Implement MapReduce program to Dept wise salary. **Data Set:** Name Age Dept Gender Salary A 23 IT Male 35000 B) Load the data into student relation  rollno,name,gpa  Load the data into department relation  rollno,deptno,deptname  Join the above two relatlions using Pig Latin | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **18** | A) Implement MapReduce based solution to find the particular employee details.  Empid, name, age, address, phone,  1001, john, 45  1002, jack, 39  1003,Bob,33  Example: find jack details.  b) Load the data into relations in Pig Latin .  stdno name deptgpa  find the details of students who got maximum, minim gpa. | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **19** | A) Implement mapreduce program to find the maximum temperature of each year from weather data.  assume weather data has year and temperature  b)Load the customers data and find country wise customer count using PigLatin  Customers: custid,custname,address, city, state, country. | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **20** | Implement MapReduce based solution to find Itemwise total Sale amount.  Itemid description saleamount  b) Load the student data and display student data based on Gpa descending order using PigLatin  Rollno ,Name, Gpa | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **21** | A) Implement MapReduce program to find count of middle age, senior and youth in the organization. (<30 youth, 30-60 middle, >60 senior) **Data Set:** Name Age Dept Gender Salary A 23 IT Male 35000 B 35 Finance Female 50000 C 29 IT Male 40000  B) Load the above empolyee data into relation using PigLatin and display all senior employees. | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **22** | A) Implement mapreduce program to find citywise student count.  dataset  student roll no Name dept  S001 Ramu IT  b) Load student data using Pig Latin, display the students of vijayawada city.  stdrollno name address city | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |
| **23** | A) Implement MapReduce program to find Dept wise employee count.  EmpnoEmpNameDept Salary  b) Load the following data into relations and join using Pig Latin  Orders: custid,itemid, itemname,orderdate,deliverydate  Customers: custid,custname,address, city, state, country. | **CO2,CO3** | **PO1,PO2,PO4,PO5,PO9** |

# Internal and External Question Paper Analysis

**Internal Question Paper Marks Distribution**

|  |  |  |
| --- | --- | --- |
| **Course Outcome** | **Marks Distribution** | **%** |
| **CO 1** | 6 | 20.00 |
| **CO 2** | 14 | 46.67 |
| **CO3** | 10 | 33.33 |
| **Total** | 30 |  |

# External Question Paper Marks Distribution

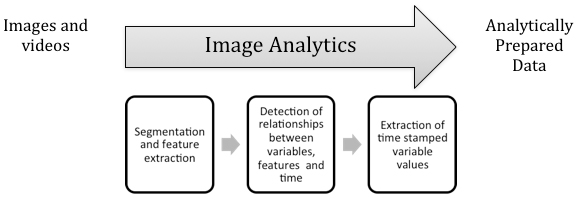
|  |  |  |
| --- | --- | --- |
| **Course Outcome** | **Marks** | **%** |
| **CO 1** | 20 | 28.57 |
| **CO 2** | 27 | 38.57 |
| **CO3** | 23 | 32.86 |
| **Total** | **70** |  |

**Project / Case Studies List**

**IMAGE PROCESSING USING THE BIG DATA TECHNOLOGY**

**The technology behind the image analytics:**

An introductory definition of image analytics is a transformation from images and videos to analytically prepared data.



**QUESTION:**

Suppose assume we have 1GB image file, we stored into the HDFS in terms of blocks. How will be process the data without losing the information (assume we have to process the image as whole, when we split the image and process, will we get the fault output)?

**ANSWER:**

If your block size is 64 MB, most probably HDFS would have split your image file into chunks and replicated it across the cluster, depending on what your cluster configuration is.

Image processing is done in hadoop using a hadoop component known as **HIPI (Hadoop Image Processing Interface)** . This component makes sure that you get the original image when you process and split image

**USECASE**

The main objective of this use case is to analyse the image processing dataset, transferring the images into hdfs using **HIPI** tool. Also finding the average pixel value of the loaded images.

**HIPI:**

* HIPI (Hadoop Image Processing Interface) is a library designed to provide efficient and high-throughput image processing in the Apache Hadoop MapReduce parallel programming framework.
* It also provides support for OpenCV.
* HIPI facilitates efficient with MapReduce style parallel programs typically executed on a cluster. It provides a solution for how to store a large collection of images on the Hadoop Distributed File System (HDFS) and make them available for efficient distributed processing.
* The latest release of HIPI has been tested with Hadoop 2.7.1.

**SETUP HADOOP:**

HIPI works with a standard installation of the Apache Hadoop Distributed File System (HDFS) and MapReduce. HIPI has been tested with Hadoop version 2.7.1.



**INSTALL GRADLE:**

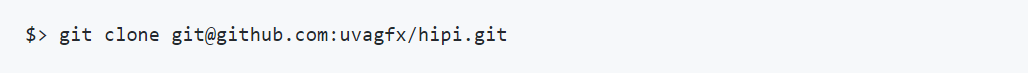
The HIPI distribution uses the Gradle build automation system for code compilation and assembly. HIPI has been tested with Gradle version 2.5:

Install gradle on your system and verify that it is in your path as well:



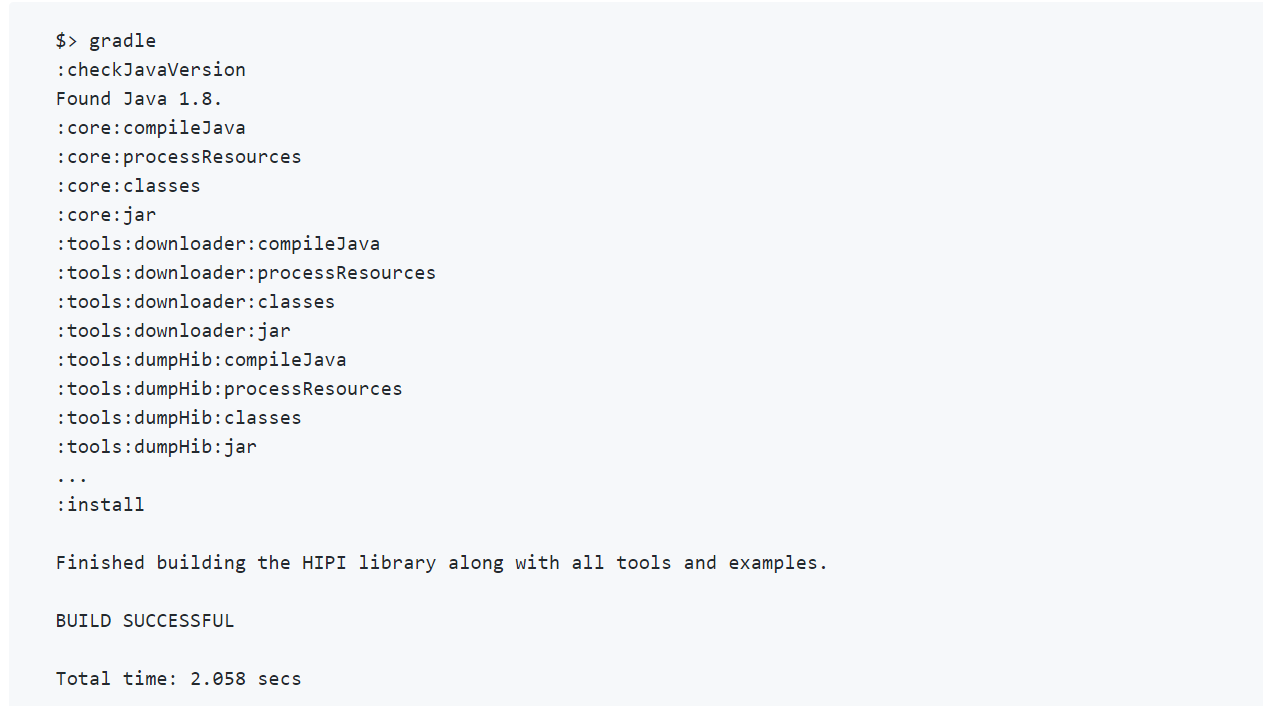
**CLONE THE HIPI REPOSITORY:**

Clone the latest HIPI distribution from GitHub:



**BUILD THE HIPI LIBRARY AND TOOLS:**

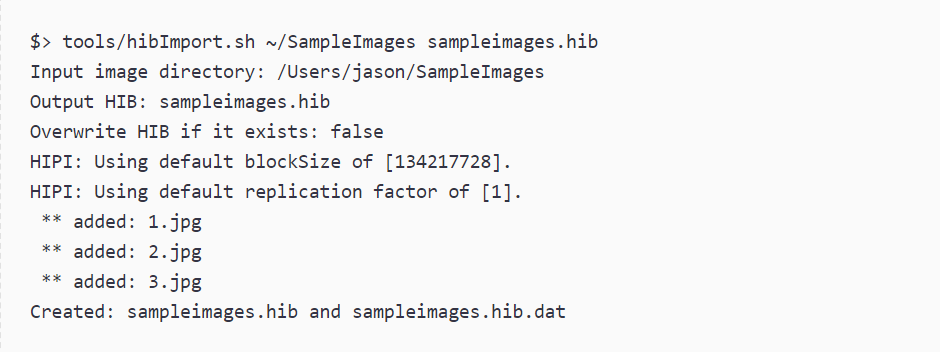
From the HIPI root directory, simply run gradle to build the HIPI library along with the complete set of tools and example programs.





**Example**

For this example, suppose the directory ~/Desktop/Sampleimages on the local file system contains four images (three JPEGs and one PNG). The following command would generate a HIB named sampleimages.hib on the HDFS consisting of this set of images:



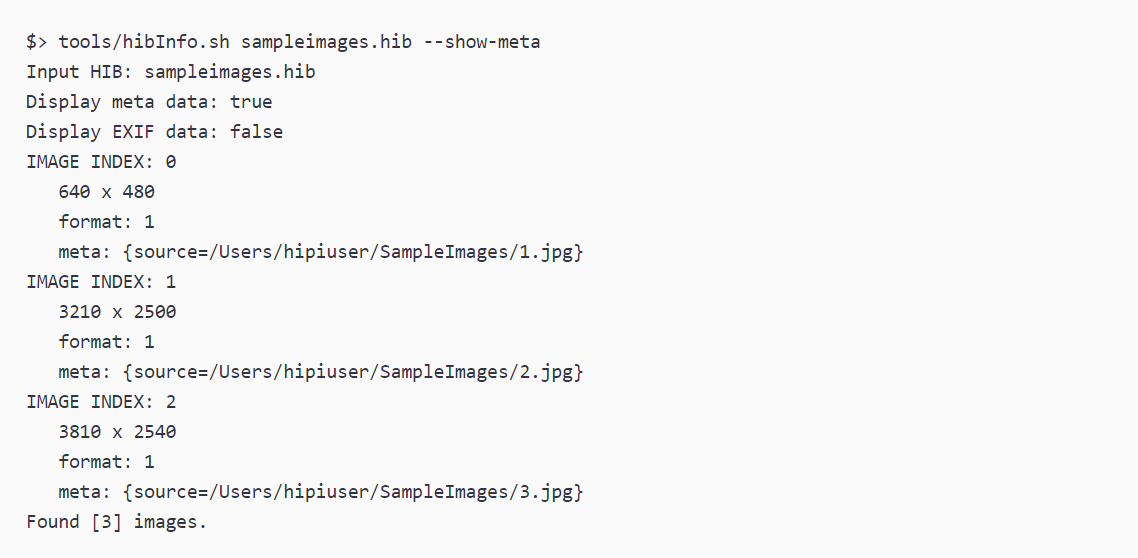
You can verify that the HDFS was updated correctly by listing the contents of the current working directory

**hibDownload**

First, we will use **hibDownload** to create a HIB from the small set of sample images. The first step is to copy a text file that lists these image URLs to a directory on the HDFS. This can be done by issuing the following commands from within the tools directory:

To make sure things worked as expected, we can inspect the contents of the text file now on the HDFS:

You can use the handy hibInfo tool that comes with HIPI to inspect the contents of this newly created HIB file:



### Computing The Average Pixel Color:

### Now let's add some actual HIPI image processing code to our program. For this example, we will be computing the average RGB value of the pixels in the images in our input HIB. Our mapper will compute the average pixel color over a single image and the reducer will add these averages together and divide by their count to compute the total average pixel color.

### Here is what our map() method looks like:

public static class HelloWorldMapper extends Mapper<HipiImageHeader, FloatImage, IntWritable, FloatImage> {

public void map(HipiImageHeader key, FloatImage value, Context context)

throws IOException, InterruptedException {

// Verify that image was properly decoded, is of sufficient size, and has three color channels (RGB)

if (value != null &&value.getWidth() > 1 &&value.getHeight() > 1 &&value.getNumBands() == 3) {

// Get dimensions of image

int w = value.getWidth();

int h = value.getHeight();

// Get pointer to image data

float[] valData = value.getData();

// Initialize 3 element array to hold RGB pixel average

float[] avgData = {0,0,0};

// Traverse image pixel data in raster-scan order and update running average

for (int j = 0; j < h; j++) {

for (int i = 0; i< w; i++) {

avgData[0] += valData[(j\*w+i)\*3+0]; // R

avgData[1] += valData[(j\*w+i)\*3+1]; // G

avgData[2] += valData[(j\*w+i)\*3+2]; // B

}

}

// Create a FloatImage to store the average value

FloatImageavg = new FloatImage(1, 1, 3, avgData);

// Divide by number of pixels in image

avg.scale(1.0f/(float)(w\*h));

// Emit record to reducer

context.write(new IntWritable(1), avg);

} // If (value != null...

} // map()

} // HelloWorldMapper

### Here is what our reducer() method looks like:

public static class HelloWorldReducer extends Reducer<IntWritable, FloatImage, IntWritable, Text> {

public void reduce(IntWritable key, Iterable<FloatImage> values, Context context)

throws IOException, InterruptedException {

// Create FloatImage object to hold final result

FloatImageavg = new FloatImage(1, 1, 3);

// Initialize a counter and iterate over IntWritable/FloatImage records from mapper

int total = 0;

for (FloatImageval : values) {

avg.add(val);

total++;

}

if (total > 0) {

// Normalize sum to obtain average

avg.scale(1.0f / total);

// Assemble final output as string

float[] avgData = avg.getData();

String result = String.format("Average pixel value: %f %f %f", avgData[0], avgData[1], avgData[2]);

// Emit output of job which will be written to HDFS

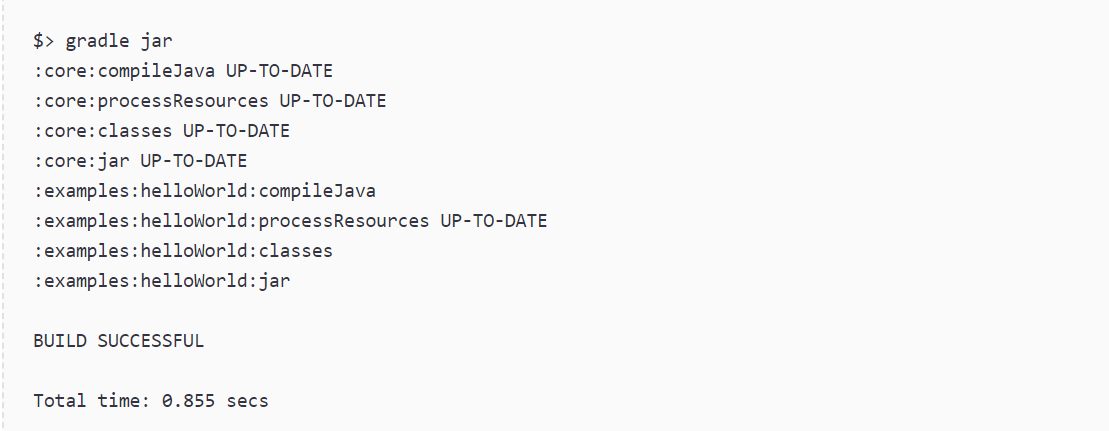
context.write(key, new Text(result));

}

} // reduce()

} // HelloWorldReducer

**Next, build helloWorld.jar and run it using the HIB we created at the beginning:**



**sampleimages\_average will contain two files:**

Whenever a MapReduce program successfully finishes, it creates the file \_SUCCESS in the output directory along with a part-r-XXXXX file for each reduce task. The average pixel value can be retrieved using the cat command:



# Experiments beyond the syllabus / projects for Advanced Learners

**1. Loading XML data into HIVE table and XML Data Processing using HIVE.**

Step 1: Create a temp hive table

create table

1. CREATE EXTERNAL TABLE companyxml(xmldata STRING) LOCATION '/user/hive/companyxml/company.xml';

created a temp table which stores  XML data as a single record. The location is the local path of the XML file.

Step 2:  Create View

load data

CREATE VIEW companyview (id, name, email, houseno, street, city, state, pincode, country, passport, visa, mobile, phone) AS SELECT

xpath(xmldata,'Company/Employee/Id/text()'),

xpath(xmldata,'Company/Employee/Name/text()'),

xpath(xmldata,'Company/Employee/Email/text()'),

xpath(xmldata,'Company/Employee/Address/HouseNo/text()'),

xpath(xmldata,'Company/Employee/Address/Street/text()'),

xpath(xmldata,'Company/Employee/Address/City/text()'),

xpath(xmldata,'Company/Employee/Address/State/text()'),

xpath(xmldata,'Company/Employee/Address/Pincode/text()'),

xpath(xmldata,'Company/Employee/Address/Country/text()'),

xpath(xmldata,'Company/Employee/Passport/text()'),

xpath(xmldata,'Company/Employee/Visa/text()'),

xpath(xmldata,'Company/Employee/Contact/Mobile/text()'),

xpath(xmldata,'Company/Employee/Contact/Phone/text()')

FROM companyxml;

This view is used to parse each tag value from the temp table.  In order to get the tag value of the XML, we can use XPath.

Step 3: Output

select data

1. SELECT \* FROM companyview;
2. ["458790","458791"]     ["Sameer","Gohar"]      ["sameer@email.com","Gohar@email.com"]  ["105","485"]      ["Grand Road","Camac Street Road"]      ["Bangalore","Mumbai"]  ["Karnataka","Maharastra"] ["560068","400001"]     ["India","India"]       ["Available","Available"] <span STYLE="color: #000000;"><strong>[]</strong></span>       ["9909999999","9908888888"]     ["8044552266"]
3. TIME taken: 0.41 seconds, Fetched: 1 ROW(s)

# Viva-Voce Questions

1. **Define Big Data and explain the Vs of Big Data.**
2. **How is Hadoop related to Big Data?**
3. **Define HDFS and YARN, and talk about their respective components.\**
4. **What do you mean by commodity hardware?**
5. **What is the purpose of the JPS command in Hadoop?**
6. **Name the different commands for starting up and shutting down Hadoop Daemons.**
7. **Why do we need Hadoop for Big Data Analytics?**
8. **Explain the different features of Hadoop.**
9. **Define the Port Numbers for NameNode, Task Tracker and Job Tracker.**
10. **What do you mean by indexing in HDFS?**
11. **What are Edge Nodes in Hadoop?**
12. **What are some of the data management tools used with Edge Nodes in Hadoop?**
13. **Explain the core methods of a Reducer.**
14. **Talk about the different tombstone markers used for deletion purposes in HBase.**
15. **How can Big Data add value to businesses?**
16. **How do you deploy a Big Data solution?**
17. **How is NFS different from HDFS?**
18. **List the different file permissions in HDFS for files or directory levels.**
19. **Elaborate on the processes that overwrite the replication factors in HDFS.**
20. **Name the three modes in which you can run Hadoop.**
21. **Explain Rack Awareness in Hadoop.**
22. **Can you recover a NameNode when it is down? If so, how?**
23. **Name the configuration parameters of a MapReduce framework.**
24. **What is a Distributed Cache? What are its benefits?**
25. **What is a SequenceFile in Hadoop?**
26. **Explain the role of a JobTracker.**
27. **Name the common input formats in Hadoop.**
28. **What is the need for Data Locality in Hadoop?**
29. **What are the steps to achieve security in Hadoop?**
30. **How can you handle missing values in Big Data?**
31. What do you know about the term “Big Data”?
32. What are the five V’s of Big Data?
33. Tell us how big data and Hadoop are related to each other.
34. How is big data analysis helpful in increasing business revenue?
35. Explain the steps to be followed to deploy a Big Data solution.
36. Define respective components of HDFS and YARN
37. What is the Command to format the NameNode?
38. Which hardware configuration is most beneficial for Hadoop jobs?
39. How to recover a NameNode when it is down?
40. What is the difference between “HDFS Block” and “Input Split”?
41. What are the limitations of MapReduce.
42. Hadoop can run Mapreduce programs written in which languages?
43. What is are subphases of Map Reduce..
44. Each phase of MapReduce has as input and output.
45. Which HDFS command is used to display the results of MapReduce program.
46. Which coordinates all the jobs run on the system by scheduling tasks.
47. The default size of HDFS block size.
48. Which is the optimizer in MapReduce Programming model.
49. For any MapReduce Job how many Job trackers are there.
50. Which Hadoop API allows to write MapReduce programs in languages other than Java.
51. What is the name of the C++ interface to Hadoop MapReduce.
52. What is used as channel in Hadoop pipes over which the tasktracker communicates with the process running the C++ Map or reduce function.
53. In which mode of Hadoop all the deamons run on the local mode.
54. Which Class providing job configuration parameters in MapReduce programming.
55. MapReduce is following which paradigm.
56. Which java deamons are responsible for MapReduce program execution.
57. Which command is used to execute MapReduce program.
58. Haoop runs the map task on a node where the input data resides in HDFS is called as.
59. Which allows us to perform a local aggregation of map output before it is transferred to reducer phase.
60. Which is reducing the amount of data that is transferred from a mapper to the reducer.
61. Which phase is there between Map and Reduce.
62. Total number of reducers is equal to what.
63. Number of input splits is determined by what.
64. The Mapper class is a generic type, with what formal type parameters.
65. Hadoop provides its own set of basic types in which package.
66. Apache Pig is an alternative for what .
67. Pig was developed as a research project at which company.
68. Features of PIG.
69. Any two usecases of Pig.
70. Pig interface is which shell.
71. What are the Pig execution modes.
72. What are the complex data types provided by Pig.
73. Which statement is used to remove duplicate tuples in Pig Latin.
74. Write pig latin statement to select tuples from a relation based on specific conditions.
75. To find the tuples of students whose gpa is greater than 4.0 from the relation Student(rollno:int,name: chararray,gpa:float). Assume the data is loaded into relation A.
76. Which pig latin statement is used to select random sample of data based on the specified sample size.
77. What is the framework used for data warehousing on top of Hadoop.
78. Hive converts SQL query into which program.
79. In HIVE table schemas are stored in which database .
80. HIVE was developed by which company to manage their evergrowing volumes of log data.
81. How HIVE is storing and analyzing the data.
82. What are the rich data types supported in HIVE.
83. What are the interfaces supported by HIV.
84. What are the different kinds of metastores in HIVE.
85. What is the default metastore for HIVE.
86. In which metastore the hive driver and the metastore interface run on different JVMS .
87. Which file format stores the data in Column oriented manner.
88. What makes it easy to run Hive commands from a wide range of programming languages.
89. For which HIVE tables when drop automatically detele metadata and its data.
90. What is the use of partitions in HIVE.
91. What is the use of buckets in HIVE.
92. In HIVE User Defined Functions must be written in which language.
93. What is the purpose of Combiner in MapReduce.
94. What is the purpose of partitioner in MapReduce.
95. What is the purpose of Distributed cache in MapReduce.
96. What are the functions of Namenode.
97. What are the functions of Datanode.
98. What are the prerequisites are required to install Hadoop.
99. What are the features of HDFS.
100. How availability is achieved by Hadoop.

**CONTINUOUS ASSESSMENT**

## Internal Assessment

|  |  |
| --- | --- |
| **Continuous Evaluation** | **Total : 30Marks** |
| Day to Day Assessment | 10 Marks |
| Program Execution | 15 Marks |
| * Writing Procedure – 5M * Execution – 5M * Viva – 5M |
| Record | 05 Marks |

**Rubrics considered in Continuous Evaluation (30M)**

|  |  |
| --- | --- |
| **Evaluation** | **Rubrics considered for Assessment** |
| Day to day Assessment – 10Marks | * Solved completely and recorded the given task   : 10Marks   * Solved and recorded upto 80% : 08-09Marks * Solved and recorded upto 70%: 06-07 Marks |
| Program Execution – 10 Marks | * Total Successful Execution : 10 Marks * Upto 80% Execution : 08-09 Marks * Upto 70% Execution : 06-07 Marks * Partial completion (upto 50-60%) : 05 Marks |
| Record – 05 Marks | * Submitted the record on time : 05 Marks * Not submitted on time : 03-04 Marks |
| Viva – 05 Marks | * Answered all questions confidently : 05Marks * Confident in concept, but couldn’t answer : 04 Marks * Partially answered due to lack of preparation :   02-03Marks |

# Internal Marks Statement

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SLNO | ROLL NO | Name | **Day to**  **Day(10)** | **Viva(**  **5)** | **Record**  **(5)** | **Procedure &**  **Execution(10)** | **Total(**  **30)** |
| 1 | **198W1A1205** | **BORUKATI SHANMUKHA SAI VARMA** | 10 | 3 | 4 | 10 | 27 |
| 2 | **198W1A1209** | **DUDDUGUNTA BHARATH REDDY** | 7 | 3 | 4 | 10 | 24 |
| 3 | **198W1A1212** | **GANESANA CHARISHMA** | 10 | 4 | 4 | 10 | 28 |
| 4 | **198W1A1214** | **GOLLA AISHWARYA** | 9 | 3 | 4 | 9 | 25 |
| 5 | **198W1A1230** | **KOLASANI SAI SRI LEKHA** | 7 | 3 | 4 | 9 | 23 |
| 6 | **198W1A1236** | **MEDASANI POOJITHA** | 9 | 1 | 4 | 9 | 23 |
| 7 | **198W1A1237** | **MEGHAVATH INDUMATHI BAI** | 8 | 3 | 4 | 8 | 23 |
| 8 | **198W1A1239** | **NAGUBANDI SAI SREYA** | 10 | 2 | 4 | 10 | 26 |
| 9 | **198W1A1245** | **PANDAMANENI CHINMAYI SUPRAJA** | 9 | 3 | 4 | 9 | 25 |
| 1  0 | **198W1A1249** | **PUNUKOLLU SURYA PRAKASH** | 9 | 4 | 5 | 9 | 27 |
| 1  1 | **198W1A1250** | **SAI LASYA TUMMALA** | 7 | 2 | 4 | 7 | 20 |
| 1  2 | **198W1A1258** | **TUNGALA THARUNI** | 8 | 4 | 5 | 8 | 25 |
| 1  3 | **198W1A1259** | **UJWALA PALADUGU** | 8 | 4 | 5 | 8 | 23 |
| 1  4 | **198W1A1260** | **VARA SWETHA** | 9 | 3 | 5 | 9 | 26 |
| 1  5 | **198W1A1261** | **VELAGA SAI SREEJA** | 9 | 5 | 5 | 9 | 28 |
| 1  6 | **198W1A1262** | **VINEELA NARRA** | 9 | 5 | 5 | 9 | 28 |
| 1  7 | **198W1A1273** | **BOYAPATI DILEEP KUMAR** | 8 | 3 | 4 | 8 | 23 |
| 1  8 | **198W1A1276** | **CHINTALAPUDI HARSHA VARDHAN** | 9 | 3 | 5 | 9 | 26 |
| 1  9 | **198W1A1282** | **HIMANSHU K SAKODE** | 9 | 4 | 4 | 9 | 26 |
| 2  0 | **198W1A1283** | **JAMPANA MEENAKSHI** | 8 | 3 | 4 | 8 | 23 |
| 2  1 | **198W1A1285** | **KADAVAKOLLU JAHNAVI** | 8 | 3 | 4 | 7 | 22 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2  2 | **198W1A1286** | **KADIYAM JYOSTHANA** | 8 | 4 | 4 | 7 | 23 |
| 2  3 | **198W1A1288** | **KELAVATH KALPANA** | 8 | 3 | 5 | 7 | 23 |
| 2  4 | **198W1A12A5** | **PIDIKITI DEEKSHITA** | 8 | 3 | 5 | 7 | 23 |
| 2  5 | **198W1A12A6** | **PRANAVI YARLAGADDA** | 9 | 4 | 5 | 9 | 27 |
| 2  6 | **198W1A12C1** | **THALAPALA CHARAN DURGA** | 7 | 3 | 5 | 7 | 22 |
| 27 | **198W1A12C8** | **YAMARTHI HARITHA** | 9 | 4 | 4 | 9 | 26 |
| 28 | **208W5A1201** | **CHINTAKAYALA PAVAN VEERA NAGENDRA KUMAR** | 9 | 5 | 4 | 9 | 25 |
| 29 | **208W5A1206** | **SHAIK FYZULLA** | 9 | 5 | 4 | 9 | 25 |

## Semester End Evaluation (70M)

|  |  |
| --- | --- |
| **External Evaluation** | **Total : 70Marks** |
| **Rubrics considered in Semester End Evaluation** |
| Procedure | **20 Marks** |
| * 100% Correctness in written procedure : 20Marks * 80-90% Correctness in written procedure : 15-19Marks * 60-70% Correctness in written procedure : 10-14Marks * 50% Correctness in written procedure : 05-10Marks |
| Program Execution | **25 Marks** |
| * Code correct and full Execution : 20-25Marks * Code nearer to solution upto some extent and partial Execution : 15-20Marks * Code incorrect to some extent with partial Execution : 10-15Marks * Code incorrect and below partial Execution : 01- 10Marks |
| Results | **10 Marks** |
| * 100% Output : 10 Marks * Upto 90% Output : 09Marks * Upto 80% Output : 08Marks * Upto 60-70% Output : 05-07Marks * Below 50% : 01-05Marks |
| Viva | 15 Marks |

**External marks Sheet**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S.No | Roll No | **Procedure** | **Program**  **Eexecution** | **Result Analysis** | **viva** | **total** |
|  |  | **20** | **25** | **10** | **15** | **70** |
|  |  |  |  |  |  |  |
| 1 | **198W1A1205** | 18 | 20 | 9 | 12 | 58 |
| 2 | **198W1A1209** | 16 | 20 | 5 | 9 | 50 |
| 3 | **198W1A1212** | 17 | 22 | 10 | 13 | 62 |
| 4 | **198W1A1214** | 10 | 15 | 10 | 10 | 45 |
| 5 | **198W1A1230** | 15 | 21 | 4 | 10 | 50 |
| 6 | **198W1A1236** | 17 | 20 | 10 | 8 | 55 |
| 7 | **198W1A1237** | 15 | 14 | 5 | 11 | 45 |
| 8 | **198W1A1239** | 15 | 20 | 5 | 7 | 47 |
| 9 | **198W1A1245** | 10 | 20 | 10 | 11 | 51 |
| 10 | **198W1A1249** | 15 | 15 | 10 | 10 | 50 |
| 11 | **198W1A1250** | 10 | 14 | 7 | 9 | 40 |
| 12 | **198W1A1258** | 13 | 20 | 10 | 12 | 55 |
| 13 | **198W1A1259** | 12 | 15 | 7 | 6 | 40 |
| 14 | **198W1A1260** | 20 | 20 | 10 | 10 | 60 |
| 15 | **198W1A1261** | 15 | 15 | 10 | 10 | 50 |
| 16 | **198W1A1262** | 17 | 18 | 10 | 11 | 56 |
| 17 | **198W1A1273** | 10 | 20 | 10 | 10 | 50 |
| 18 | **198W1A1276** | 19 | 20 | 10 | 11 | 60 |
| 19 | **198W1A1282** | 15 | 17 | 10 | 8 | 50 |
| 20 | **198W1A1283** | 15 | 15 | 8 | 8 | 46 |
| 21 | **198W1A1285** | 15 | 15 | 8 | 10 | 50 |
| 22 | **198W1A1286** | 19 | 15 | 10 | 11 | 55 |
| 23 | **198W1A1288** | 12 | 24 | 10 | 11 | 57 |
| 24 | **198W1A12A5** | 14 | 15 | 8 | 8 | 45 |
| 25 | **198W1A12A6** | 16 | 24 | 9 | 11 | 60 |
| 26 | **198W1A12C1** | 11 | 19 | 6 | 9 | 45 |
| 27 | **198W1A12C8** | 11 | 19 | 9 | 11 | 50 |
| 28 | **208W5A1201** | 18 | 24 | 9 | 13 | 65 |
| 29 | **208W5A1206** | 19 | 24 | 10 | 13 | 66 |