Experiment No:

Date:

Aim: Prerequisities for the Big Data Analysis Lab

To demonstrate the experiments in BDA we require

i) Cloudexa Lab Account

ii) Installing winscp for file transfer

iii) Install Ellipse Java IDE

Steps to create Cloudex Lab Account:-

1) Firstly open browser and search for the cloudex lab.

2) Click on the link "www.cloudexlab.com".

3) After that it loads the cloudx lab webpage, now click login with google account.

4) After signing in it asks for access to permissions grant them.

5) We should select the trail version which is initially Rs 59/- using online payment that is refunded within ten days of successful transaction.

6) The trail version expires after 7 days which can be enrolled again.

7) To execute the lab program we need to open the profile and click on "MyLab" which open the constraints in it select "webconsole".

8) To get access to webconsole we have to give the constraints of username and password in it.

9) After the entering of Loginid and password your id related webconsole command id is represented.

10) Now you can execute the commands related to the lab in the console.

Steps to install winscp:-

1) To install we need to open browser and search the link "www.winscp.net/eng/download"

2) After opening the link we get the download link for winscp version 5.17.9 (10.6MB)

3) After downloading the winscp setup file we need to install it.

4) Click the install for all users option which is recommended

5) Accept the licence agreement, then click next for installation and after it is installed click on the finish button which launches the winscp.

6) Now give the access to the webconsole through the login credentials by clicking on new session.

7) Copy the link address of webconsole and paste in the address which change the SFTP option

8) Change the option from that to SFTP and give the login credentials of cloudex lab.

9) Now click on login button which opens the remote server and helps the transferring of the files into cloudex by just drag and drop.

10) For the transfer it asks continue the process click on yes then the file are transferred to cloudex successfully.

Steps to install Eclipse:

1) To install Eclipse first search for the link "www.eclipse.org" in your browser.

2) In that select the download button then click on Download x84.64 which moves the downloading section.

3) After dowloading the Eclipse we need to install that .exe file in desktop/laptop

4) Click on the exe file and accept for the installation process

5) Click on next and finish button appears by clicking it the eclipse is launched.

6) After the Eclipse is launched we need to select the first option "Ellipse IDE Developers" which open the workspace

7) After opening it we need to select a new java project by clicking on new button.

8) After we need to enter the code and execute related to the programming aspects.

9) This is the final step of Eclipse that is needed for the libraries for the code execution or converting the jar file.

Experiment No:

Date:

Aim :- Practice on basic Linux commands

File commands :-

1) ls :- It is used for listing directory

Syntax :- $ls [options] [file/direc]

Output :- Display all files in directory.

2) ls -al :- It is used for formatted listing with hidden files.

3) ls -lt :- It is used for sorting the formatted listing by time modifications

4) cd dir :- It changes directory from one to other

Syntax :- $cd directory name

Example :- $cd bda

opens the directory bda

5) cd :- It changes any directory and comes to home directory

Syntax :- $cd

Ex :- $ cd

6) pwd :- shows the current working directory

Syntax :- $pwd

Example :- $pwd

o/p :- nagaakhil

7) mkdir dir :- It creates a new directory

Syntax :- $mkdir [directoryname]

Ex :- $mkdir count

8) cat > file :- display the standard input in a file

syntax :- $cat > filename

Example :- $ cat > file1

hi
hello
hru

9) more file :- output the contents of file

10) head file :- output the first 10 lines of a file

Syntax :- $head filename

Ex :- $head file1

11) tail file :- output the last 10 lines of a file

Syntax :- $tail filename

Ex :- $ tail file1

12) tail -f file:- Output the contents of file as it grows, starting with the last 10 lines

13) touch file :- create or update file

syntax :- $touch filename.

14) rm file :- removes a file

syntax :- $rm filename

15) rm -r dir :- Deleting a directory

syntax :- $rm -r directoryname

16) rm -f file :- Force to remove the file

syntax :- $rm -f filename

17) rm -rf dir :- force to remove directory dir

syntax :- $rm -rf directoryname

18) cp file1 file2 :- copy the contents in file1 to file2

syntax :- $ cp filename filename

19) cp -r dir1 dir2 :- copy the directory1 file to directory2 creates the directory2 if not exist

syntax :- $ cp -r directoryname directoryname

20) mv file1 file2 :- Rename or move file1 to file2

Syntax :- $mv filename filename.

Process Management commands:

1) Ps :- To display the currently working process

Syntax :- $ps [options]

Ex :- $ps

2) top :- Display all running processes

Syntax :- $top

3) Kill pid :- Kill the process with given pid

Syntax :- $kill pid

file permission commands :-

1) chmod octal file :- change the permission of file to octal which can be found seperated for user, group, world by adding

- 4 - read(r)
- 2 - write(w)
- 1 - execute(x)

Searching command

1) grep pattern file, search for pattern in file

Syntax :- $grep pattern filename

System Info :-

1) date :- show the current date and time

syntax :- $ date [options]

Ex :- $ date

Wed 16 Dec 2020 04:00:01 PM

2) cal :- Shows the month's calender

syntax :- $ cal [month] [year]

3) uptime :- shows current uptime

syntax :- $ uptime [option]

4) who :- Display who users are online

syntax :- $ who [option]

5) whoami :- the username and logged in time of you

syntax :- $ whoami

6) man :- This command helps to attain information of a particular command

Syntax :- $ man commandname

Example :- $ man ls

Experiment No:

Date:

Aim:- Implement the following file management tasks in hadoop

i) Adding files and directories

ii) Retrieving files

iii) Deleting files.

Hadoop provides a set of command line utilities that work similarly to the linux file commands

Default directories:-

Local file system: /home/cloudera/ HDFs: /user/cloudera

Basic file commands :-

Hadoop file command take the form of

hadoop fs -cmd <args>

where cmd is specific file command

<args> is variable no. of. arguments

Example:- command for listing files is

hadoop fs -ls

Now, we check the adding files, retrieving files and deleting files

9) Adding files and directories:- Before running hadoop programs need to put the data into HDFS first

1) mkdir:- create a directory in HDFS at given path

hadoop fs -mkdir <path>

Ex:- hadoop fs -mkdir/usr/cloudera/folder1

(or)

hadoop fs -mkdir. myfolder

Ex:- hadoop fs -mkdir/usr/cloudera/bda/hadoop

2) ls:- List the contents of a directory

hadoop fs -ls <args>

Ex:- hadoop fs -ls

3) put or copyfromLocal:- upload a file in HDFS

hadoop fs -copyfromlocal localsrc dst

Copy single src file or multiple files from local file system to hadoop distributed file system.

Example:- Create two file in local filesystem using cat or using any editor nano or gedit

Cat > file1

This is Hadoop Lab

cat > file2

This is Bigdata Lab

hadoop fs -put file1 /usr/cloudera/folder1

hadoop fs -copyfromLocal file2 /usr/cloudera/folder1

hadoop fs -put file3

checking :- hadoop fs -lsr /usr/cloudera/folder1

hadoop fs -ls

b) Retriving files :- Copy files from HDFS to local system.

(1) Dowload : get or copyToLocal

copies/downloads files to the local file system

hadoop fs -get hdfs-src localdst

(or)

hadoop fs -copyToLocal hdfs-src localdst

Ex:-

hadoop fs -get /usr/claudera/folder1

hadoop fs -copyToLocal /usr/cloudera/folder1/file1

Another way to access the data is to display it we can use the hadoop file command with unix pipes to send its O/P for further processing.

hadoop fs -cat file1

hadoop fs -cat file1/head

hadoop fs -tail file1

3) Deleting files :- Hadoop command for removing for removing file is rm

Example:-

hadoop fs -rm file1

hadoop fs -rmr myfolder

Looking up help :- A list of hadoop file commands together with the usage and description of each command can see by using help command.

hadoop fs -help cmd

Example :- hadoop fs -help ls

Experiment No:

Date:

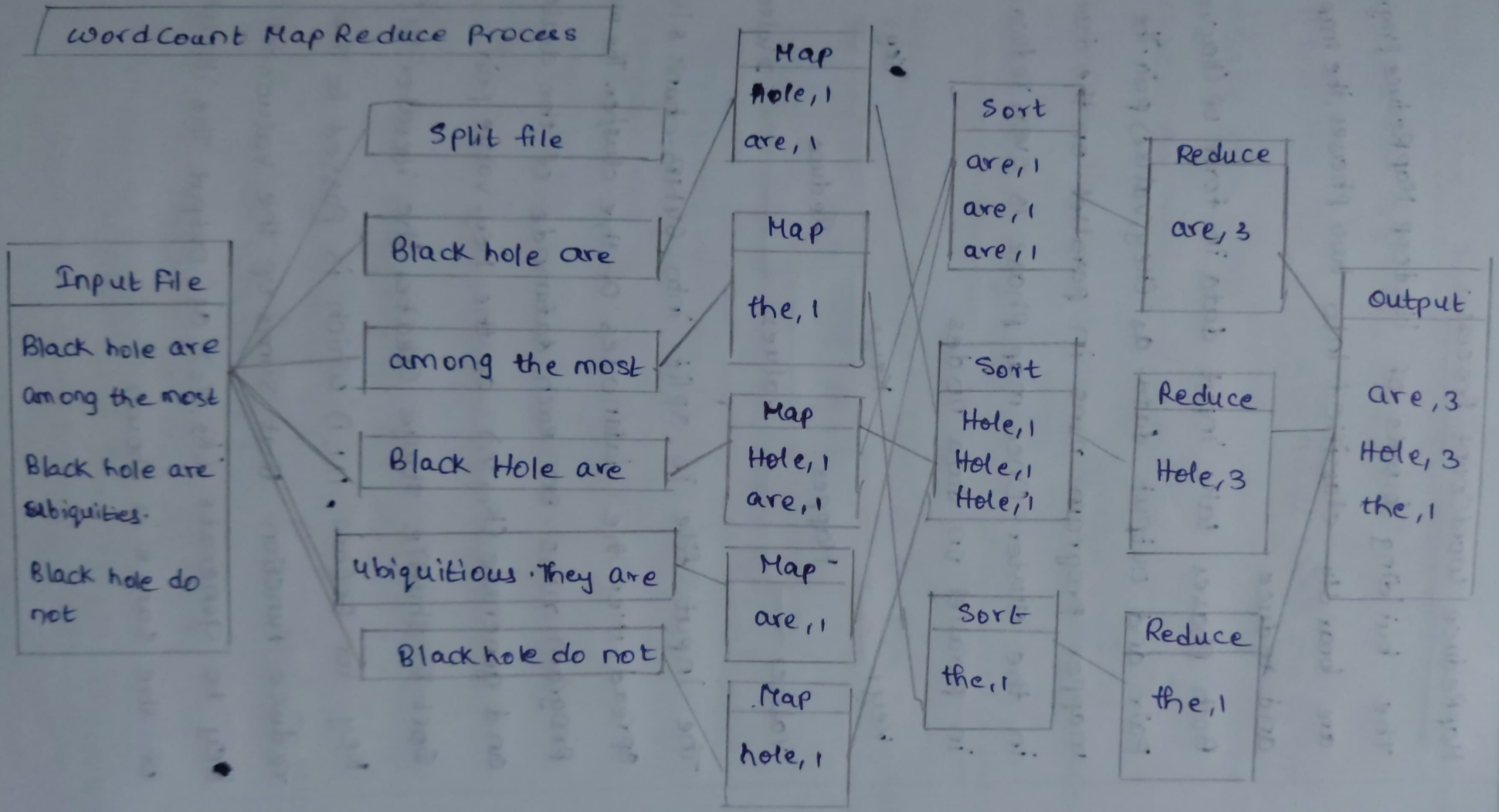
Aim :- Run a basic word count MapReduce program to understand MapReduce paradigm.

Description :- Hadoop MapReduce is a system for parallel processing which was initially adopted by Google for executing the set of functions over large data sets in batch mode which is stored in the fault-tolerant large cluster.

The input dataset which can be a terabyte file broken down into chunks of 64MB by default is the input to Mapper function. The Mapper function then filters and sort these data chunks on hadoop cluster data nodes based on the business requirement. After the distributed computation is completed, the output of the mapper function is parsed to reducer function which combines all the elements back together to provide the resulting output.

Example of hadoop MapReduce usage is "word count" algorithm in raw java using classes provided by Hadoop libraries. To begin, consider below figure which breaks the word-count process into steps.

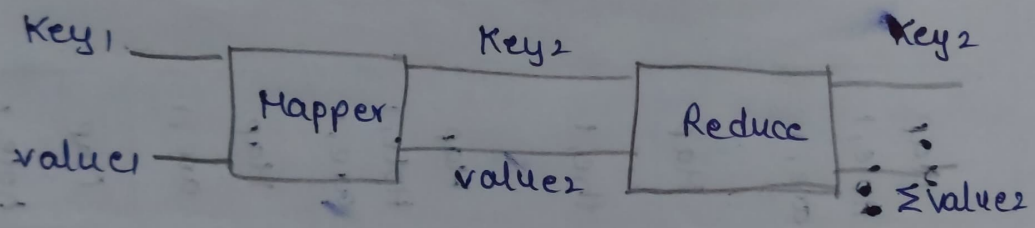
WordCount Map Reduce Process



MapReduce WordCount Process :-

The building blocks of Hadoop MapReduce programs are boardly classified into two phases, the map and reduce

Both phases take input data in form of (key, value) pair and output data as (key, value) pair. The mapper program runs in parallel on the data nodes in the cluster. Once map phase is over, reducer run in parallel on data nodes



The input file is split into 64MB chunk & is spread over the datanodes of the cluster. The mapper program runs on each datanode of the cluster and generate (K_1, V_1) as the key-value pair

Sort & shuffle stage creates the iterator for each key for Ex (are, 1, 1, 1) which is passed to the reduce function that sums up the values for each key to generate (K_2, V_2) as output. The illustration of the same is shown

Hadoop MapReduce Algorithm for word count program

1) Take one line at a time

2) split the line into individual word one by one

3) Take each word

4) Note the frequency count for each word

5) Report the list of words and frequency tabulation

Hadoop MapReduce code structure for word count:

Step1:- Import Hadoop libraries for path, Configuration, IIO, MapReduce and utilities

```
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
```

Step2:- The summary of the classes defined in the word count map reduce program is as below:

```
Public class WordCount {
  Public static class Map Extends MapperC
  {                    'LongWritable, Text, Text, IntWritable>
    ..........
  }
```

```
Public static class Reduce extends Reducer
        <Text, IntWritable, Text, IntWritable>
{
    - - - - -
    - - - -
}
public static void main(String args[]) throws
  Exception {
  - - - - -
  - - - - -
}
```

step3:- Define the map class. The key and value input pair have to be serializable by the framework and hence need to implement the writable interface

Output pairs donot need to be of the same types of input pairs. output pairs are collected with calls to context. Inside the static class "map" we are declaring an object with the name "one" to store the inumental value of the given word and the particular word is stored in the variable named 'word".

step4:- Reduce class will accept shuffled key-value pairs as input. the code then totals the value for the key-value pairs with the same key &

output the totaled key-value pairs.

Steps: The main method sets up the MapReduce configuration by defining the type of input. In this case, the input is text. The code then defines the Map, combine and Reduce classes as well as specifying the input/output formats.

Steps: The full java code for the "coord count" program is as follows:

```
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
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.mapreduce.lib.Output.
 TextOutputFormat;
```

```
Public class WordCount
{
   Public static class Map Extends Mapper<long
    Writable, Text, Text, IntWritable>
   {
    Private final static IntWritable one = new
    IntWritable();
    Private Text word = new Text();
    Public void map (LongWritable key, Text value,
    Context context) throws IOException,
    InterruptedException
    {
      String line = value.toString();
      StringTokenizer tokenize = new StringTokenizer(line)
      while (tokenizer.hasMoreToken()) {
          word.set(tokenizer.nextToken());
          context.write(word, one);
      }
    }
}
Public static class Reduce Extends Reducer <Text,
 IntWritable, Text, IntWritable>
{
```

```
public void reduce(Text key, Iterable<Int
Writable> values, Context context) throws
IOException, InterruptedException
{
  int sum = 0;
  for(IntWritable val : values)
  {
    sum += val.get();
  }
  Context.write(key, new IntWritable(sum));
 }
}
Public static void main(String[] args) throws
Exception
{
  Configuration conf = new Configuration();
  Job job = new Job(conf, "wordcount");
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  job.setMapperClass(Map.class);
  job.setReducerClass(Reduce.class);
```

```
    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.waitForCompletion(true);
  }
}
```

Step7:- Execute the code

i) write/create an input text file

```
$vi text
hello how are you
i am fine
hello who are you
```

ii) put it to the user server

```
$hadoop fs -put/home/nagaakhil1519/t.text/usr/
  nagaakhil1519
$hadoop fs -cat/usr/nagaakhil1519/t.txt
hello how are you
i am fine
hello who are you
```

iii) Now execute the code and observe output

```
$hadoop jar wo1.jar com.sample.WordCountJob
user/nagaakhil1519/t.txt /usr/nagaakhil1519/output
```

iv) See the output

```
$hadoop fs -ls /user/nagaakhil1519/output
found 3 items
rw-r--r-- 3 /user/nagaakhil1519/output/_SUCCESS
rw-r--r-- 3 /user/nagaakhil1519/output/part-r-00000
rw-r--r-- 3 /user/nagaakhil1519/output/part-r-00001
$hadoop fs -cat/user/nagaakhil1519/output/part-r-00000
i 2
you 2
$hadoop fs -cat/user/nagaakhil1519/output/part-r-00001
am 2
are 2
fine 1
hello 2
how 1
who 1
```