Exercise - 1

Aim:- Write the installation steps for Hadoop in ubuntu.

Hadoop 2.6.1 installation in ubuntu 16.04:-

1) Installing Java.

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

it will install java source in your machine at /usr/lib/jvm/java-7-openjdk-i386.

To check java version.

```
$ java -version.
```

2) ADD GROUP & USER.

```
$ sudo addgroup hadoop
$ sudo adduser --ingroup hadoop hduser
$ sudo adduser hduser sudo
$ sudo gedit /etc/sudoers.
```

in this file at the line "# Allow members of group sudo to execute any command" Add the below line

```
% hduser ALL=(ALL:ALL) ALL
```

LOGOUT the current USER, LOGIN as HDUSER, the below given steps are performed by only HDUSER.

3) Installing SSH

```
$ sudo apt-get install openssh-server
```

Configuring SSH

```
$ Sudo su hduser
$ ssh-Keygen
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys.
```

Start ssh.

```
$ sudo /etc/init.d/ssh restart
```

Test your Connectivity : $ ssh localhost

4) Disabling IPV6

Since hadoop doesn't work on IPV6, we should disable it. One of another reason is also that it has been developed and tested on IPV4 stacks. Hadoop nodes will be able to communicate if we are having IPV4 cluster.

For getting your IPV6 disable in your Linux machine, you need to update /etc/sysctl.conf by

```
$ sudo gedit /etc/sysctl.conf
```

adding following line of codes at end of the file.

```
# disable ipv6
  net.ipv6.conf.all.disable_ipv6 = 1
  net.ipv6.conf.default.disable_ipv6 = 1
  net.ipv6.conf.lo.disable_ipv6 = 1
```

```
$ cat/proc/sys/net/ipv6/conf/all/disable_ipv6.
```

it gives "zero" (if system not restart). after restart the result is "one" (ipv6 is disable)

Tip:- You can use nano, gedit, and vi editor for updating all text files for this configuration purpose.

5) Download latest Apache Hadoop source from Apache mirrors.

* First you need to download Apache Hadoop 2.6.1 (i.e. hadoop-2.6.1.tar.gz) or latest version source from Apache download Mirrors. You can also try stable hadoop to get all latest features as well as recent bugs solved with Hadoop source, choose location where you want to place all your hadoop installation, I have chosen /usr/local/hadoop.

```
## Locate to hadoop installation parent dir
 $hduser@pingax:~$ cd <your hadoop directory>
## Extract hadoop source
 $ sudo tar -xvzf hadoop-2.6.1.tar.gz
## Move hadoop-2.6.1 to hadoop folder.
 $ sudo mv hadoop-2.6.1 /usr/local/hadoop.
## Assign ownership of this folder to Hadoop
 user.
 $ sudo chown -R hduser:hadoop /usr/local/hadoop.
 $ sudo chmod -R 777 /usr/local/hadoop.
```

create Hadoop temp directories for namenode and Datanode.

```
$ Sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenod
$ Sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode
```

Again assign ownership of this Hadoop temp folder to Hadoop user.

```
$ Sudo chown -R hduser:hadoop /usr/local/hadoop-tmp/
$ Sudo chmod -R 777 /usr/local/hadoop-tmp/
```

6) update Hadoop Configuration files

User profile : update $HOME/.bashrc

```
$ sudo gedit ~/.bahrc.
```

update hduser configuration file by appending the following environment variables at the end of this file.

```
# -- HADOOP ENVIRONMENT VARIABLES START --#
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386
export PATH=$PATH:$JAVA_HOME/bin
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME = $HADOOP_HOME.
export YARN_HOME = $HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR = $HADOOP_
       HOME/lib/native.
export HADOOP_OPTS = "-Djava. library. path = $
          HADOOP_HOME/lib"

# -- HADOOP ENVIRONMENT VARIABLES END -- #
```

Configuration file : hadoop-env.sh.

To edit, file, fire the below given command

~~hduser@pllab~~/usr/local/hadoop/etc/hadoop $ sudo gedit hadoop-env.sh

update JAVA_HOME variable,

```
JAVA_HOME = /usr/lib/jvm/java-7-openjdk-i386.
```

Configuration file : core-site.xml.

To edit file, fire the below given command

```
/usr/local/hadoop/etc/hadoop $ sudo gedit core-site.xml
```

Paste these lines into < configurations tag.

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

Configuration file : hdfs-site.xml

To edit file, fire the below given command

```
/usr/local/hadoop/etc/hadoop $ sudo gedit hdfs=site.xml
```

paste these lines into <Configuration> tag.

```
<property>
    <name> dfs.replication</name>
     <value> 1 </value>
</property>
<property>
     <name> dfs.namenode.name.dir</name>
      <value> file:/usr/local/hadoop_tmp/hdfs/
                namenode</value>
</property>
 <property>
      <name> dfs.datanode.data.dir</name>
      <value> file:/usr/local/hadoop_tmp/hdfs/
                 datanode</value>
 </property>
```

Configuration file : yarn-site.xml

To edit file, fire the below given command

```
/usr/local/hadoop/etc/hadoop $ sudo gedit yarn-
                                     site.xml
```

paste these lines into <configuration> tag.

```
<property>
```

```
  <name> yarn. nodemanager. aux-services</name>
  <value> mapreduce_ shuffle </value>
</property>
<property>
   <name> yarn. nodemanager. aux-services. mapre
          -duce. shuffle. class </name>
   <value> org. apache. hadoop. mapred. shuffle
          . Handler </value>
</property>
```

Configuration file : mapred-site. xml

```
## Copy template of mapred-site. xml. template
   file.
$. cp/usr/local/hadoop/etc/hadoop/mapred-site.xml.
   template
   /usr/local/hadoop/etc/hadoop/mapred-site.xml
## To edit file, fire the below given command.
   /usr/local/hadoop/etc/hadoop $ sudo gedit mapred
                      -site.xml
## paste these lines into <configuration> tag
   <property>
      <name> mapreduce. framework. name </name>
      <value> yarn </value>
   </property>
```

Format Namenode

hdfs namenode -format.

Start all Hadoop daemons.

Start hdfs daemons:

```
/usr/local/hadoop$ start-dfs.sh.
```

· Start MapReduce daemons:

```
/usr/local/hadoop $ start-yarn.sh.
```

Instead ~~of~~ of both of these above Command you can also use start-all.sh, but its now deprecated so its not recommended to be used for better Hadoop operations.

1) Track/Monitor/verify.

verify Hadoop daemons:

```
jps.
```

In terminal, when we start doing hadoop programs then must and should we have to give the following two commands.

① start-all.sh.

② jps.

After giving this command, if we get the following 6 nodes, it is successfully installed

JPS
SecondaryNameNode
DataNode
Resource Manager
NodeManager
NameNode.

Experiment - 2

Aim :- To implement Word Count Map Reduce program using standalone hadoop without combiner.

Description:-

* 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 Summari-ze 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.
* It is used to reduce the volume of data transfer b/w map & reduce.
* The input for this program is a text file contains n-lines of text as input
* The following command is used to run the word count application by taking input files from the input directory.

→ hadoop jar wc.jar prgmname input_dir output_dir

to verify resultant files in o/p folder.

→ hadoop fs -ls output_dir

to see the output.

→ hadoop fs -cat output_dir/part-r-00000.

* The output is in the form of key-value pair as.

<word, count>

* ~~out~~

* If we don't use combiner class combiner count of input and output is represented as '0'.

Program:-

word Count.java:-

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.
                        File Input Format;
import org.apache.hadoop.mapreduce.lib.output
                        FileOutput Format;

Public class Word Count
{
    public static class TokenizerMapper extends
        Mapper<object, Text, Text, IntWritable>
    {
```

```
Private final static IntWritable one = new
          IntWritable(1);
private Text word = new Text();
public void map(Object key, Text value, Context
                    context) throws IOException,
                 InterruptedException
  {
    StringTokenizer itr = new StringTokenizer
                        (value.toString());
    while (itr.hasMoreTokens())
     {
        word.set (itr.nextToken());
        context.write (word, one);
      }
  }
}
public static class IntSumReducer extends
       Reducer< Text, IntWritable, Text, IntWritable>
{
  private IntWritable result = new IntWritable();
  public void reduce (Text key, Iterable< IntWritable>
                values, Context context)
       throws IOException, InterruptedException
   {  int sum =0;
      for ( IntWritable val : values)
       {   Sum+=val.get();
       }
```

```
        resut. set (sum);
        Context. write (key, result);
    }
}
public static void main(String[] args) throws
                                Exception
{
    Configuration conf = new Configuration();
    Job job = Job. get Instance (conf, "word count");
    job. set Jor ByClass (WordCount. class);
    job. set MapperClass (TokenizerMapper. class);
    job. setReducerClass (IntSumReducer. class);
    job. setOutputKeyClass (Text. class);
    job. setOutputValueClass (IntWritable. class);
    FileInputFormat. addInputPath (job, new
                                Path (args[0]));
    FileOutputFormat. setOutputPath (job, new Path(
                                args[1]));
    System. exit (job. waitForCompletion (true) ? 0 : 1);
  }
}
```

a(input Text file):-

This is hadoop lab
today is wednesday
This is important lab
this is used for analyze big data Concepts
in this we use map reduce

we have to do this

we have to strictly follow the class.

this is important lab

This is important lab

this is used for analyze big data concepts.

in this we use map reduce.

we have to do this

we have to strictly follow the class

this is important lab.

Output:-

```
This      3
analyze    2
big        2
class      2
concepts   2
data       2
do         2
follow     2
for        2
hadoop     1
have       4
important    4
in          2
is          8
lab         5
map         2
reduce      2
```

strictly	2
the	2
this	8
to	4
today	1
use	2
used	2
we	6
wednesday	1.

Experiment-3

Aim:- To implement WordCount Map Reduce Program using standalone hadoop with combiner class.

Description:-

* combiner class is used in b/w the Map class and the Reduce class to reduce the volume of data transfer b/w map and Reduce.

* usually, the output of the map task is large and the data transferred to the reduce task is high.

* combiner class main task is, it uses group by key and groups the <key, value> pairs of input from map like (k_1:v , k_2:v_1, k_1:v_2) to grouping as (k_1:v,v_2 , k_2:v_1) gives this to Reducer.

* Its main task is to produce summary information from a large dataset because it replaces the original Map output.

* The input for this program is a text file contains n-lines of text as input (a.txt)

* The following commands used in execution

```
hadoop jar k1.jar.prgmname ip_dir op_dir

hadoop fs -ls op_dir

hadoop fs -cat op_dir/part-r-00000
```

* In MapReduce we have Record Reader, Record writer.

* Recordwriter~~'s~~ ~~output~~ takes input as <key, value> pair and gives the output as text.

Program:-

word Count1.

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.
                        FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.
                        FileOutputFormat;

public class WordCount1
{
  public static class TokenizerMapper extends
      Mapper<object, Text, Text, IntWritable>
   {
    private final static IntWritable one = new
                   IntWritable(1);
    private Text word = new Text();
    ~~public voi~~
Public void map (Object Key, Text value, Content
            content) throws IOException, Interrupted
                  Exception
{
   StringTokenizer itr = new StringTokenizer
                        (value.toString());
    while (itr.hasMoreTokens())
     {   word.set (itr.nextToken());
          context.write (word, one);
      }
   }
}
public static class IntSumReducer extends
      Reducer< Text, IntWritable, Text, IntWritable>
{
   private IntWritable result = new IntWritable();
   public void reduce (Text Key, Iterable<IntWritable
             values, Content context) throws
             IOException, InterruptedException
   {
        int sum=0;
        for (IntWritable val : values)
        {
            sum += val.get();
        }
        result.set(sum);
        context.write (Key, result);
   }
}
```

```
public static void main(String[] args) throws
                Exception
{
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    Job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path
                        (args[0]));
    FileOutputFormat.setOutputPath(job, new Path
                        (args[1]));
    System.exit(job.waitForCompletion(true)?0:1);
  }
}
```

Input:- a.txt. (same. as in previous experiment)

Output:-

```
This      3
analyze   2
big       2
class     2
Concepts  2
data      2
do        2
follow    2
```

```
for          2
hadoop       1
have         4
important    4
in           2
is           8
lab          5
map          2
reduce       2
strictly     2
the          2
this         8
to           4
today        1
use          2
used         2
we           6
wednesday    1 .
```