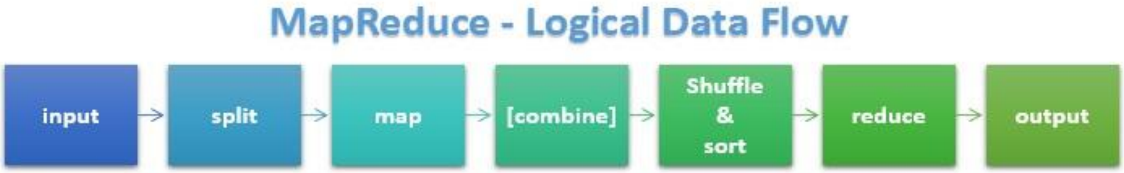
Lab Manual

Big Data Analytics Lab



S.SURYANARAYANARAJU

Department of Computer Science and Engineering
SRKR Engineering College (A), Bhimavaram, India

Course objectives:

1. Able to run the tools like UBUNTU Operating System, Java 8, and Eclipse.
2. Able to create Hadoop Environment and develop Map-Reduce Programs for Real time applications.

Course outcomes:

1. Create Hadoop environment.
2. Develop a solution for a given problem using map reduce.

LIST OF PROGRAMS

Experiment #1

Setting up Hadoop on Pseudo distributed mode (Single node cluster).

Objective:: *Setting up Hadoop on Pseudo distributed mode*

The pseudo-distributed mode is also known as a single-node cluster where both NameNode and DataNode will reside on the same machine.

In pseudo-distributed mode, all the Hadoop daemons will be running on a single node. Such configuration is mainly used while testing when we don't need to think about the resources and other users sharing the resource.

In this architecture, a separate JVM is spawned for every Hadoop components as they could communicate across network sockets, effectively producing a fully functioning and optimized mini-cluster on a single host.

PROGRAM MODULE:

Step 1: Install Java 8.

Install Java 8 (Recommended Oracle Java) Hadoop requires a working Java 1.5+ installation. However, using Java 8 is recommended for running Hadoop.

1.1*Install Python Software Properties*

Command: **sudo apt-get install python-software-properties**

1.2 Add Repository

Command: **sudo add-apt-repository ppa:webupd8team/java**

1.3 Update the source list

Command :**sudo apt-get update**

1.4 Install Java

Command: **sudo apt-get install oracle-java8-installer**

Step 2: Configure SSH

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus your local machine if you want to use Hadoop on it.

2.1 Install Open SSH Server-Client

Command: **sudo apt-get install openssh-server openssh-client**

2.2 Generate KeyPairs

Command: **ssh-keygen -t rsa -P ""**

2.3 Configure password-less SSH

Command: **cat $HOME/.ssh/id_rsa.pub >> HOME/.ssh/authorized_keys**

2.4 Check by SSH to localhost

Command: **ssh localhost**

Step 3: Install Hadoop

3.1 Download Hadoop

Command:**Wgethttp://archive.cloudera.com/cdh5/cdh/5/hadoop-2.5.0-dh5.3.2.tar.gz**

3.2 Untar Tar ball

Command: **tar xzf hadoop-2.5.0-cdh5.3.2.tar.gz**

Step 4: Setup Configuration

4.1 Edit .bashrc

Edit .bashrc file located in user‟s home directory and add following parameters.

Command :**vi .bashrc**

```
export HADOOP_PREFIX="/home/cse/hadoop-2.5.0-cdh5.3.2"
export PATH=$PATH:$HADOOP_PREFIX/bin
export PATH=$PATH:$HADOOP_PREFIX/sbin
export HADOOP_MAPRED_HOME=${HADOOP_PREFIX}
export HADOOP_COMMON_HOME=${HADOOP_PREFIX}
export HADOOP_HDFS_HOME=${HADOOP_PREFIX}
export YARN_HOME=${HADOOP_PREFIX}
```

Command: **source .bashrc**

4.2 Edit hadoop-env.sh

hadoop-env.sh contains the environment variables that are used in the scriptto run Hadoop like Java home path, etc. Edit configuration file hadoop-env.sh(located in HADOOP_HOME/etc/hadoop) and set JAVA_HOME.

Command: **vi hadoop–env.sh**export JAVA_HOME=/usr/lib/jvm/java-8-oracle/

4.3 Edit core-site.xml

core-site.xml informs Hadoop daemon where NameNode runs in the cluster.It contains configuration settings of Hadoop core such as I/O settings that arecommon to HDFS & MapReduce. Edit configuration file core-site.xml (located inHADOOP_HOME/etc/hadoop) and add following entries.

Command :**vi core-site.xml**

```
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost:9000</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>/home/cse/hdata</value>
</property>
</configuration>
```

Note

/home/cse/hdata is a sample location; please specify a location where you have ReadWrite privileges.

4.4 Edit hdfs-site.xml

hdfs-site.xml contains configuration settings of HDFS daemons (i.e.NameNode, DataNode, Secondary NameNode). It also includes the replication factorand block size of HDFS. Edit configuration file hdfs-site.xml (located inHADOOP_HOME/etc/hadoop) and add following entries.

Command: vi **hdfs-site.xml**

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

4.5 Edit mapred-site.xml

mapred-site.xml contains configuration settings of MapReduce applicationlike number of JVM that can run in parallel, the size of the mapper and the reducerprocess, CPU cores available for a process, etc.In some cases, mapred-site.xml fileis not available. So, we have to create the mapred-site.xml file using mapred-site.xmltemplate. Edit configuration file mapred-site.xml (located in HADOOP_HOME/etc/hadoop) and add following entries.

Command:**vi mapred-site.xml**

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

4.6 Edit yarn-site.xml

yarn-site.xmlcontains configuration settings of ResourceManager andNodeManager like application memory management size, the operation needed on program & algorithm, etc.Edit configuration file mapred-site.xml (located in HADOOP_HOME/etc/hadoop) and add following entries.

Command:**vi yarn-site.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

Step 5: Start the Cluster

5.1 Format the name node:

Command: **bin/hdfs namenode -format**

5.2 Start HDFS Services

Command: **sbin/start-dfs.sh**

5.3 Start YARN Services

Command: sbin**/start-yarn.sh**

5.4 Check whether services have been started

To check that all the Hadoop services are up and running, run the below command.
Command: **jps**

NameNode

DataNode

ResourceManager

NodeManager

SecondaryNameNode

Step 6. Stop the Cluster

6.1 Stop HDFS Services

Command:**sbin/stop-dfs.sh**

6.2 Stop YARN Services

Command: sbin**/stop-yarn.sh**

Experiment # 2

Setting up Hadoop on multi-node

Objective *::Setting up Hadoop on distributed cluster environment*

In Fully Distributed Mode, the daemons Name Node, Job Tracker, SecondaryNameNode (Optional and can be run on a separate node) run on the Master Node. The daemons Data Node and Task Tracker run on the Slave Node.

PROGRAM MODULE:

Step 1. Add Entries in hosts file: Edit hosts file and add entries of **both master and slaves.**

sudo vi /etc/hosts

MASTER-IP master

SLAVE01-IP slave01

SLAVE02-IP slave02

(NOTE: In place of MASTER-IP, SLAVE01-IP, SLAVE02-IP put the value of the corresponding IP).

Example

192.168.1.190 master

192.168.1.191 slave01

192.168.1.195 slave02

Step 2. Install Java 8 (Recommended Oracle Java)

Hadoop requires a working Java 1.5+ installation. However, using Java 8 is recommended for running Hadoop.

2.1 Install Python Software Properties

Command :**sudo apt-get install python-software-properties**

2.2 Add Repository

Command :**sudo add-apt-repository ppa:webupd8team/java**

2.3 Update the source list

Command :**sudo apt-get update**

2.4 Install Java

Command :**sudo apt-get install oracle-java8-installer**

Step 3. Configure SSH

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus your local machine if you want to use Hadoop on it.

3.1 Install Open SSH Server-Client

Command :**sudo apt-get install openssh-server openssh-client**

3.2 Generate KeyPairs

Command :**ssh-keygen -t rsa -P ""**

3.3 Configure password-less SSH

3.3.1 Copy the generated ssh key to master node's authorized keys.

Command: **cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys**

3.3.2 Copy the master node's ssh key to slave's authorized keys.

Command:

```
ssh-copy-id -i $HOME/.ssh/id_rsa.pub cse@slave01
ssh-copy-id -i $HOME/.ssh/id_rsa.pub cse@slave02
```

3.4 Check by SSH to all the Slaves

```
ssh slave01
ssh slave02
```

Step 4. Install Hadoop

1 Download Hadoop

```
wget http://archive.cloudera.com/cdh5/cdh/5/hadoop-2.5.0-cdh5.3.2.tar.gz
```

2 Untar Tar ball

Command : tar xzf hadoop-2.5.0-cdh5.3.2.tar.gz

Step 5. Hadoop multi-node cluster setup Configuration

1.Edit .bashrc

Edit .bashrc file located in user"s home directory and add following parameters.

Command : vi .bashrc

```
export HADOOP_PREFIX="/home/cse/hadoop-2.5.0-cdh5.3.2"
export PATH=$PATH:$HADOOP_PREFIX/bin
export PATH=$PATH:$HADOOP_PREFIX/sbin
export HADOOP_MAPRED_HOME=${HADOOP_PREFIX}
export HADOOP_COMMON_HOME=${HADOOP_PREFIX}
export HADOOP_HDFS_HOME=${HADOOP_PREFIX}
export YARN_HOME=${HADOOP_PREFIX}
```

Command : source .bashrc

2.Edit hadoop-env.sh

hadoop-env.shcontains the environment variables that are used in the scripttorun Hadoop like Java home path, etc. Edit configuration file hadoop-env.sh(located in HADOOP_HOME/etc/hadoop) and set JAVA_HOME.

Command :**vi hadoop–env.shexport JAVA_HOME=/usr/lib/jvm/java-8-oracle/**

3.Edit core-site.xml

core-site.xmlinforms Hadoop daemon where NameNode runs in the cluster. Itcontains configuration settings of Hadoop core such as I/O settings that arecommon to HDFS & MapReduce.Edit configuration file core-site.xml (located in HADOOP_HOME/etc/hadoop)and add following entries.

Command : vi core-site.xml

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

```
<property>
<name>hadoop.tmp.dir</name>
<value>/home/cse/hdata</value>
</property>
</configuration>
```

Note: Here /home/ashok/hdata is a sample location; please specify a location where you have Read Write privileges.

4.Edit hdfs-site.xml

hdfs-site.xmlcontains configuration settings of HDFS daemons (i.e.NameNode, DataNode, Secondary NameNode). It also includes thereplication factor and block size of HDFS.Edit configuration file hdfs-site.xml (located in HADOOP_HOME/etc/hadoop)

and add following entries

Command : vi hdfs-site.xml

```
<configuration>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
</configuration>
```

5.Edit mapred-site.xml

mapred-site.xmlcontains configuration settings of MapReduce applicationlike number of JVM that can run in parallel, the size of the mapper andthe reducer process, CPU cores available for a process, etc.In some cases, mapred-site.xml file is not available. So, we have to create themapred- site.xml file using mapred-site.xml template. Edit configurationfile mapred-site.xml (located in HADOOP_HOME/ etc/hadoop) and add followingentries.

Command : vi mapred-site.xml

```
<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>
```

6.Edit yarn-site.xml

yarn-site.xmlcontains configuration settings of ResourceManager andNodeManager like application memory management size, the operationneeded on program & algorithm, etc.

Edit configuration file mapred-site.xml(located in HADOOP_HOME/etc/hadoop) and add following entries

Command : vi yarn-site.xml

```
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
<name>yarn.resourcemanager.resource-tracker.address</name>
<value>master:8025</value>
</property>
<property>
<name>yarn.resourcemanager.scheduler.address</name>
<value>master:8030</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
```

```
<value>master:8040</value>
</property>
</configuration>
```

7. Edit salves

Edit configuration file slaves (located in HADOOP_HOME/etc/hadoop) and addfollowing entries:

```
slave01

slave02
```

Now Hadoop is set up on Master, now setup Hadoop on all the Slaves.

Install Hadoop On Slaves.

Step 6. Setup Prerequisites on all the slaves

Run following steps on all the slaves

1. Add Entries in hosts file

2. Install Java 8 (Recommended Oracle Java)

Step 7. Copy configured setups from master to all the slaves

7.1. Create tarball of configured setup

Command :**tar czf hadoop-2.5.0-cdh5.3.2 .tar.gz hadoop-2.5.0-cdh5.3.2**

(NOTE: Run this command on Master)

7.2. Copy the configured tarball on all the slaves

Command :**scp hadoop.tar.gz slave01:**

(NOTE: Run this command on Master)

Command :**scp hadoop.tar.gz slave02:**

(NOTE: Run this command on Master)

7.3. Un-tar configured Hadoop setup on all the slaves

Command :**tar xvzf hadoop.tar.gz**

(NOTE: Run this command on all the slaves)

Now Hadoop is set up on all the Slaves. Now Start the Cluster.

Step 8. Start the Hadoop Cluster

Let us now learn how to start Hadoop cluster?

8.1. Format the name node

Command :**bin/hdfs namenode -format**

(Note1: Run this command on Master

(Note2: This activity should be done once when you install Hadoop, else it will delete all the data from HDFS)

8.2. Start HDFS Services

Command :**sbin/start-dfs.sh**

(Note: Run this command on Master)

8.3. Start YARN Services

Command :**sbin/start-yarn.sh**

(Note: Run this command on Master)

8.4. Check for Hadoop services

8.4.1. Check daemons on Master

Command :**jps**

NameNode

ResourceManager

8.4.2. Check daemons on Slaves

Command :**jps**

DataNode

NodeManager

Step 9. Stop The Hadoop Cluster

Let us now see how to stop the Hadoop cluster?

9.1. Stop YARN Services

Command :**sbin/stop-yarn.sh**

(Note: Run this command on Master)

9.2. Stop HDFS Services

Command :**sbin/stop-dfs.sh**

(Note: Run this command on Master)

Experiment # 3

Basic Hadoop file system commands.

Objective :: Basic commands in Hadoop file sysem

PROGRAM MODULE:

1. Create a directory: **hdfs dfs -mkdir /foldername**

   example:**hdfs dfs -mkdir /input**

2. Create a text file: **vi filename.extension**

   example:**vi sample.txt**

3. Upload into HDFS : **hdfs dfs -put „/local file path/filename.extension" /hdfsfoldername**

   example:**hdfs dfs -put '/home/cse/sample.txt' /input**

4. Download HDFS to local:**hdfs dfs -get '/hdfs path of a file/filen.extension'/localfoldername**

   example:**hdfs dfs -get '/inp/sample.txt' home/cse**

5. To display the content of a file : **hdfs dfs -cat /path of a file/filename.extension**

   example:**hdfs dfs -cat input/sample.txt**

6. To display first few lines of a file: **hdfs dfs -cat /path of a file/filename.extension |head-number**

   example:**hdfs dfs -cat /inp/sampe.txt |head -5**

7. To display last few lines of a file: **hdfs dfs -cat /path of a file /filename.extension |tail -number**

example:**hdfs dfs -cat /inp/sampe.txt |tail -10**

8. Remove a file or directory: **hdfs dfs -rm /path of a file or directory**

   example:hdfs dfs -rm np/sample.txt

9. Copy a file from source to destination: **hdfs dfs -cp sourcefile.extension destinationfolder**

   example: **hdfs dfs – cp home/cse/sample.txt home/desktop**

Experiment # 4

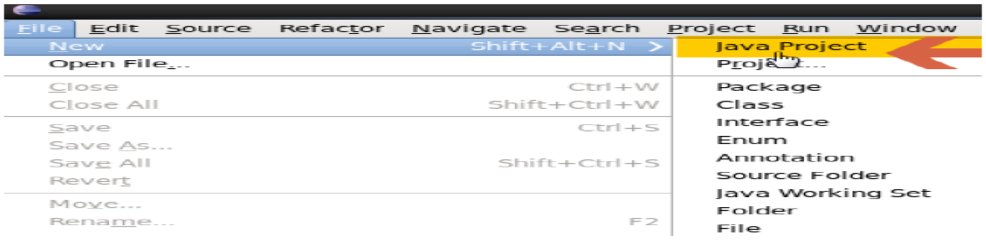
Write a map-reduce program for word count.

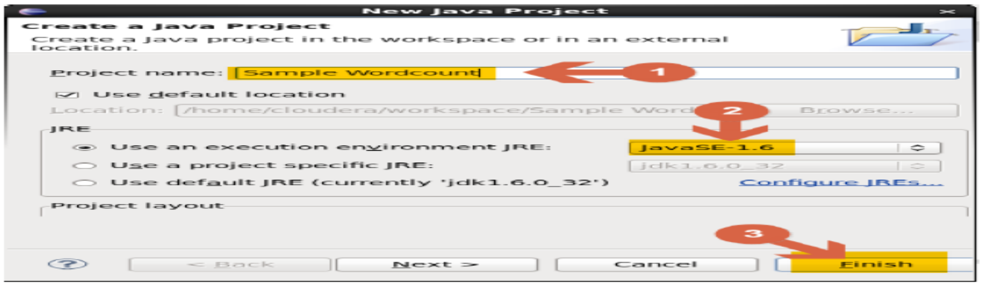
Objective :: *Simple word count program in Map reduce environment*

Running the WordCount Example in Hadoop MapReduce using Java Project with Eclipse Now, let‟s create the WordCount java project with eclipse IDE for Hadoop. Even if you are working on Cloudera VM, creating the Java project can be applied to any environment.

Step 1 –

Let‟s create the java project with the name "Sample WordCount" as shown below - File > New > Project > Java Project > Next. "Sample WordCount" as our project name and click "Finish":




New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: Sample Wordcount

☑ Use default location

Location: /home/cloudera/workspace/Sample Word

JRE

● Use an execution environment JRE: JavaSE-1.6

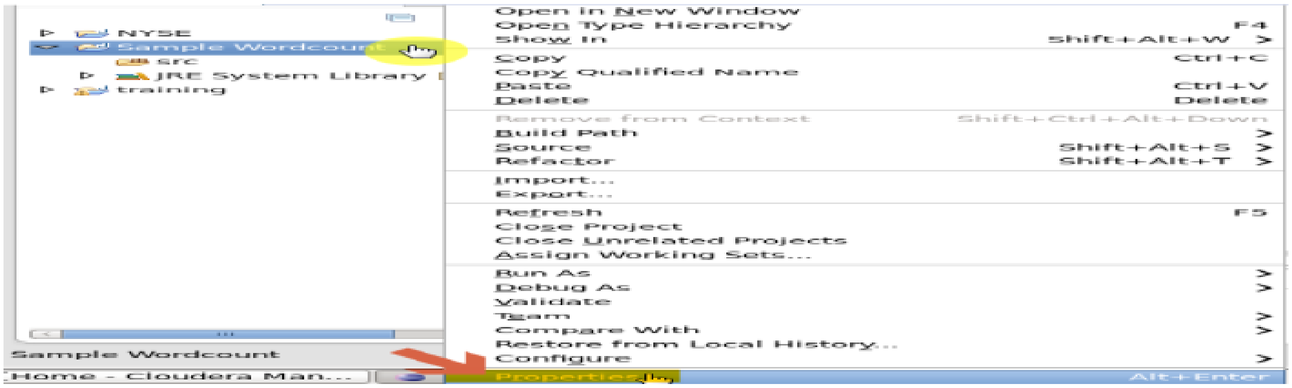
○ Use a project specific JRE: jdk1.6.0_32

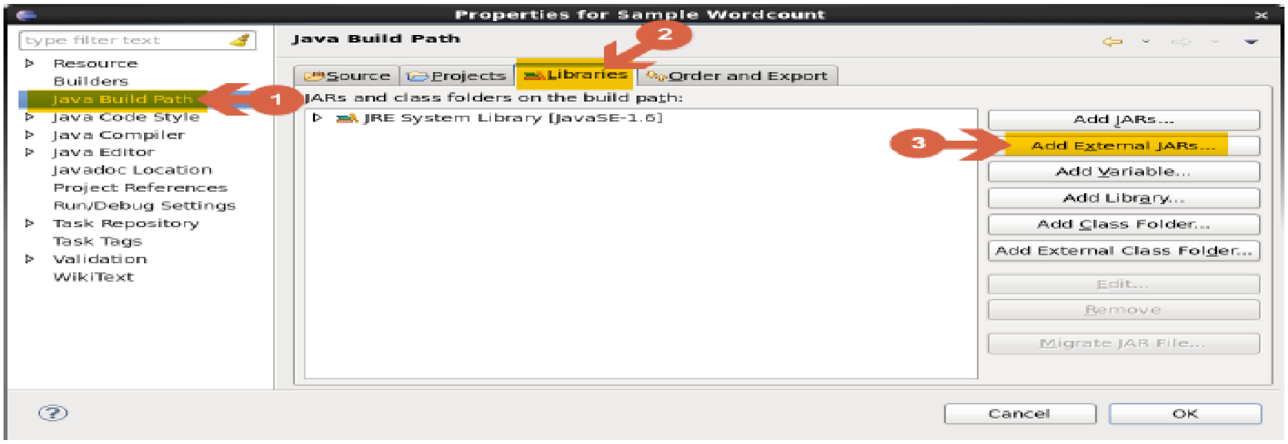
○ Use default JRE (currently 'jdk1.6.0_32')

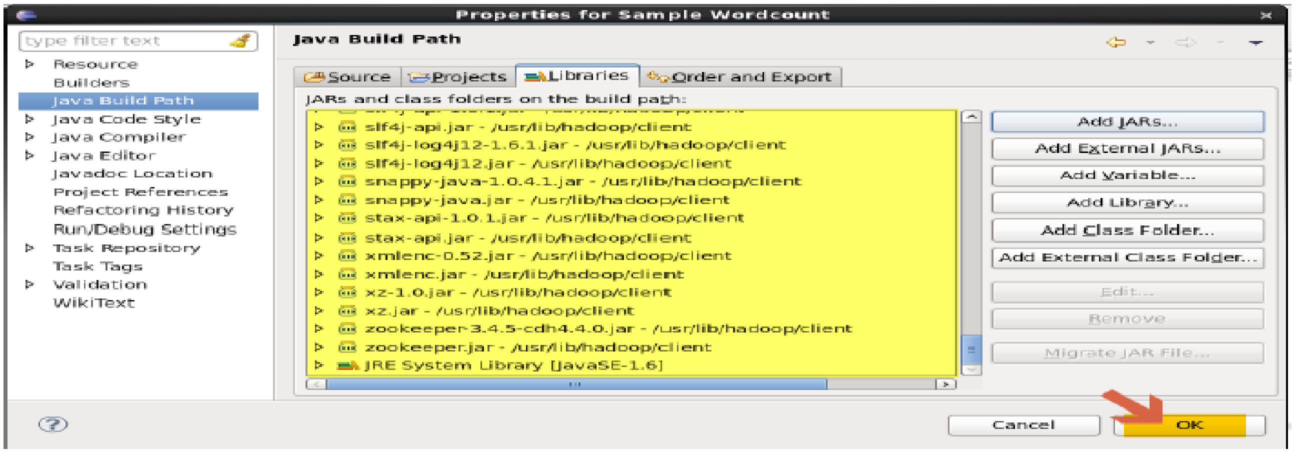
Project layout

Step 2 -

The next step is to get references to hadoop libraries by clicking on Add JARS as follows



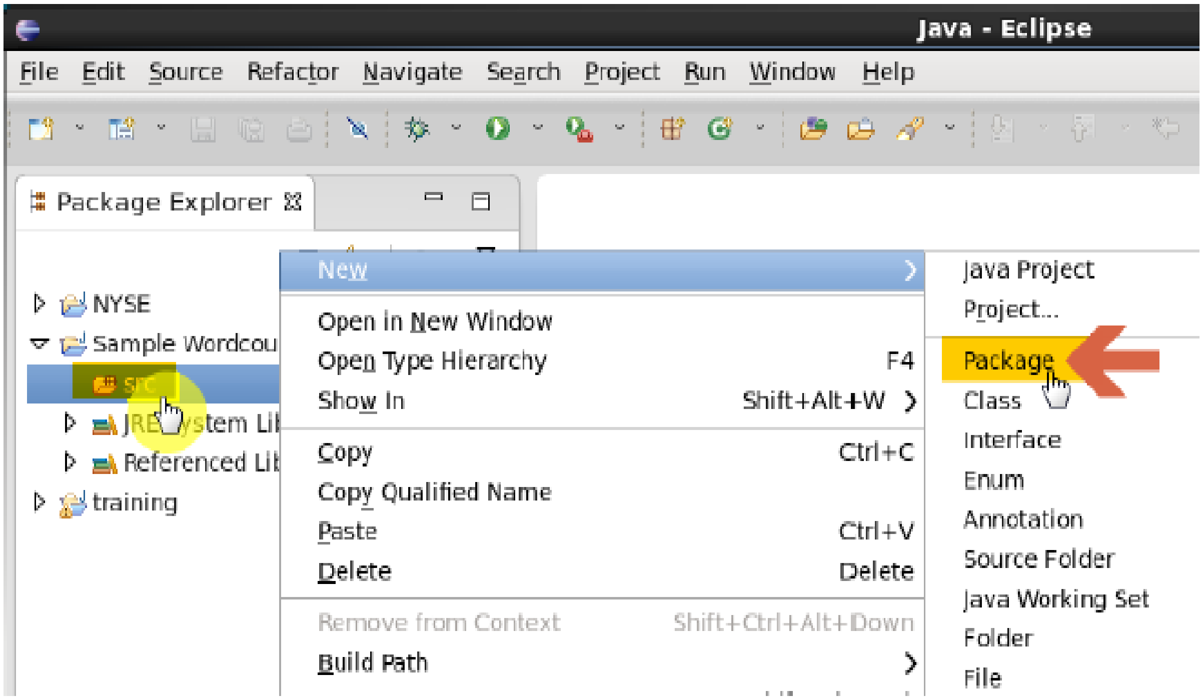


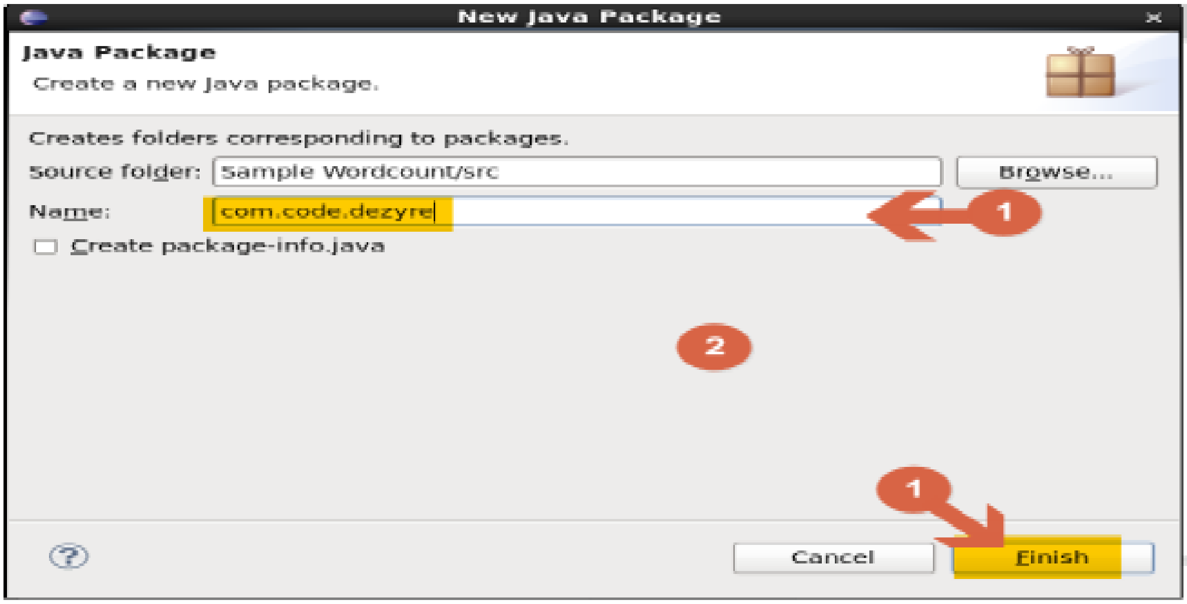


slf4j-api.jar - /usr/lib/hadoop/client
slf4j-log4j12-1.6.1.jar - /usr/lib/hadoop/client
slf4j-log4j12.jar - /usr/lib/hadoop/client
snappy-java-1.0.4.1.jar - /usr/lib/hadoop/client
snappy-java.jar - /usr/lib/hadoop/client
stax-api-1.0.1.jar - /usr/lib/hadoop/client
stax-api.jar - /usr/lib/hadoop/client
xmlenc-0.52.jar - /usr/lib/hadoop/client
xmlenc.jar - /usr/lib/hadoop/client
xz-1.0.jar - /usr/lib/hadoop/client
xz.jar - /usr/lib/hadoop/client
zookeeper-3.4.5-cdh4.4.0.jar - /usr/lib/hadoop/client
zookeeper.jar - /usr/lib/hadoop/client
JRE System Library [JavaSE-1.6]

Step 3 -

Create a new package within the project with the name com.code.dezyre-





Java Package

Create a new Java package.

Creates folders corresponding to packages.

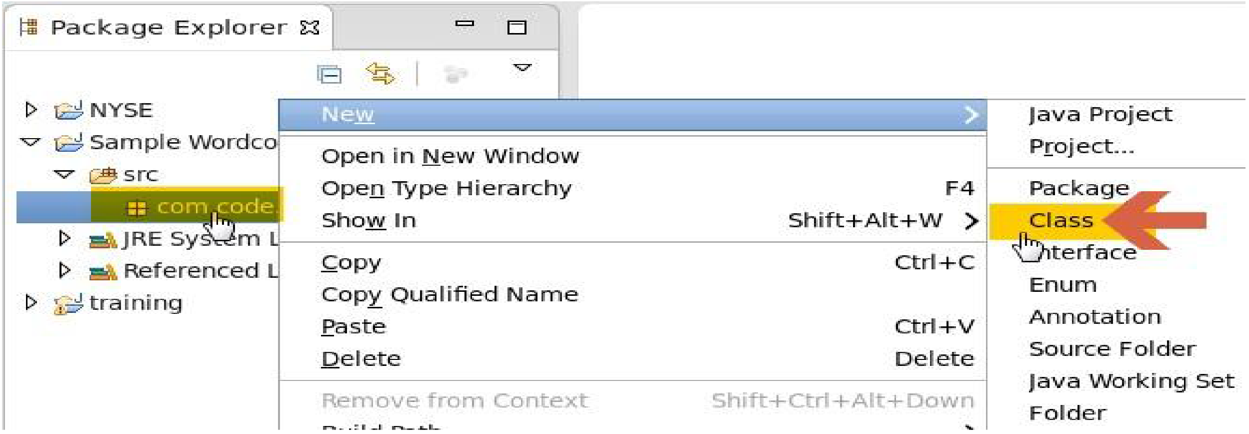
Source folder: Sample Wordcount/src

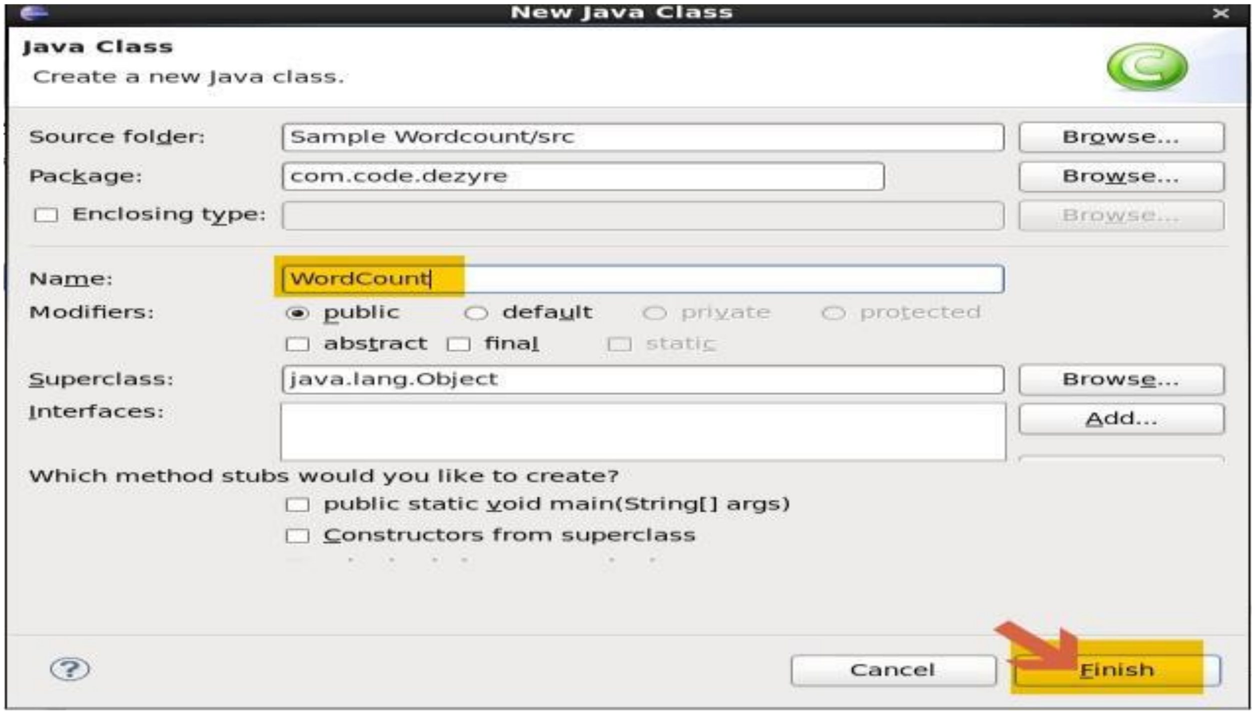
Name: com.code.dezyre

☐ Create package-info.java

Step 4 –

Now let‟s implement the WordCount example program by creating a WordCount class under the project com.code.dezyre.





Source folder: Sample Wordcount/src

Package: com.code.dezyre

□ Enclosing type:

Name: WordCount

Modifiers: ● public ○ default ○ private ○ protected

□ abstract □ final □ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

□ public static void main(String[] args)

□ Constructors from superclass

Step 5 -

Create a Mapper class within the WordCount class which extends MapReduceBase Class toimplement mapper interface. The mapper class will contain –

1. Code to implement "map" method.
2. Code for implementing the mapper-stage business logic should be written withinthis method.

PROGRAM MODULE:

Mapper Class Code for WordCount (**Mapper.java**)

```
package wordcount;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class Mapper 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);
}  } }
```

Step 6 –

Create a Reducer class within the WordCount class extending MapReduceBase Class to implement reducer interface. The reducer class for the wordcount example in hadoop will contain the –

1. Code to implement "reduce" method
2. Code for implementing the reducer-stage business logic should be written within this method.

Reducer Class Code for WordCount (Reducer.java)

```
package wordcount;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class Reducer 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();
}
result.set(sum);
context.write(key, result);
}
}
```

Step 7 –

Create main() method within the WordCount class and set the following properties

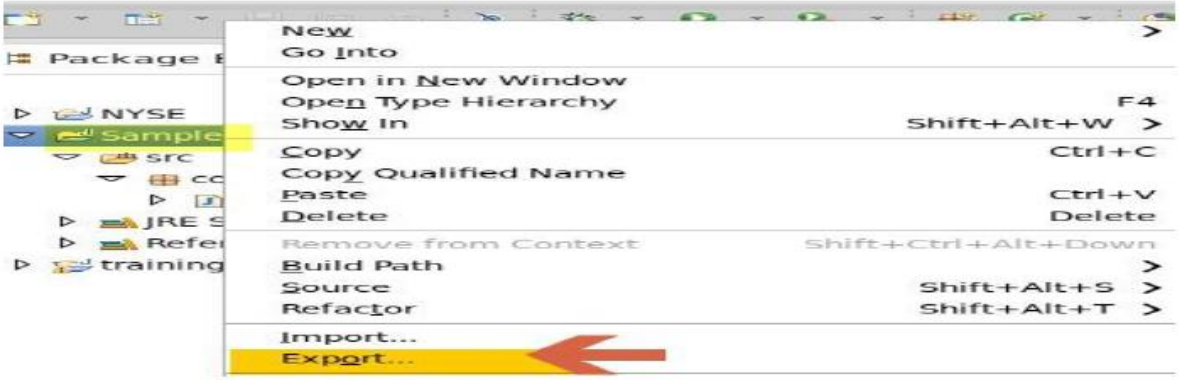
1. OutputKeyClass
2. OutputValueClass
3. Mapper Class
4. Reducer Class
5. InputFormat
6. OutputFormat
7. InputFilePath
8. OutputFolderPath

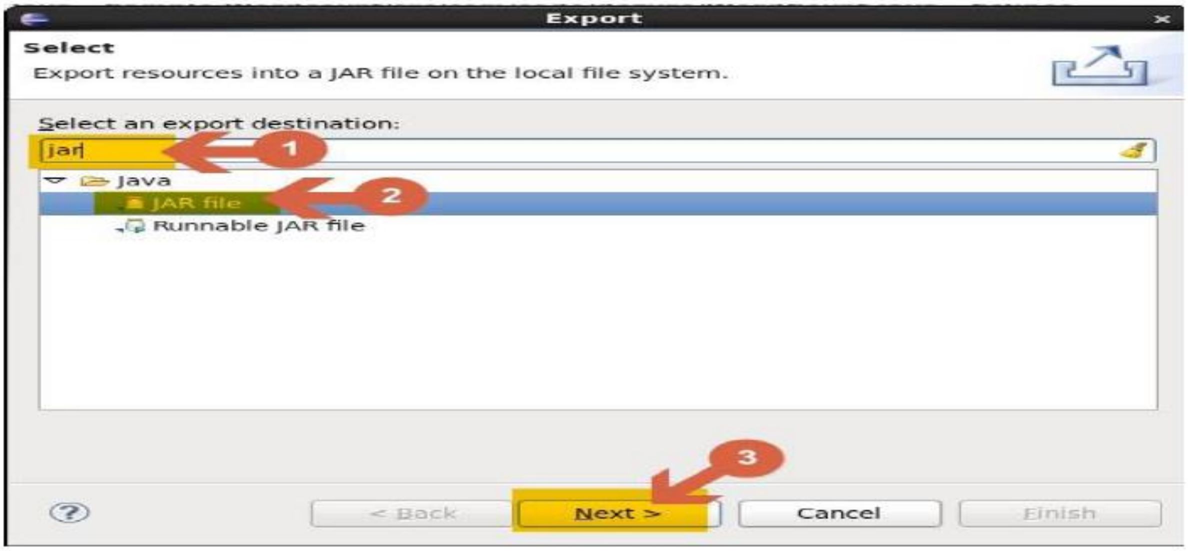
Driver Class Code for WordCount (WordCount.java)

```
package wordcount;
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.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class WordCount {
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
String[] otherArgs = new GenericOptionsParser(conf,
args).getRemainingArgs();
if (otherArgs.length < 2) {
System.err.println("Usage: wordcount <in> [<in>...] <out>");
System.exit(2);
}
Job job = Job.getInstance(conf, "word count");
job.setJarByClass(WordCount.class);
job.setMapperClass(Mapper.class);
job.setReducerClass(Reducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
for (int i = 0; i < otherArgs.length - 1; ++i) {
FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
}
FileOutputFormat.setOutputPath(job, new Path(otherArgs[otherArgs.length - 1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

Step 8 –

Create the JAR file for the wordcount class –

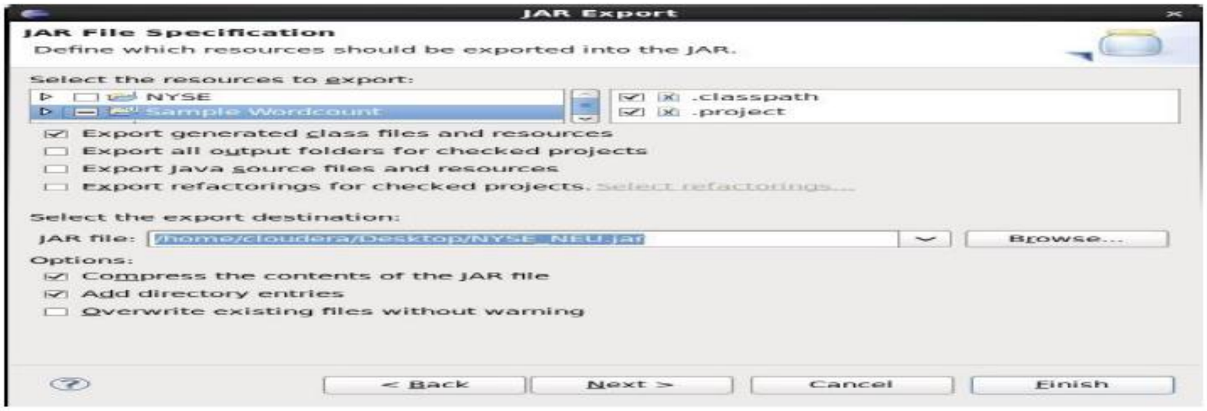



Export

Select

Export resources into a JAR file on the local file system.

Select an export destination: jar


JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- ▷ □ NYSE
- ▷ ⊟ Sample Wordcount
- ☑ ☒ .classpath
- ☑ ☒ .project

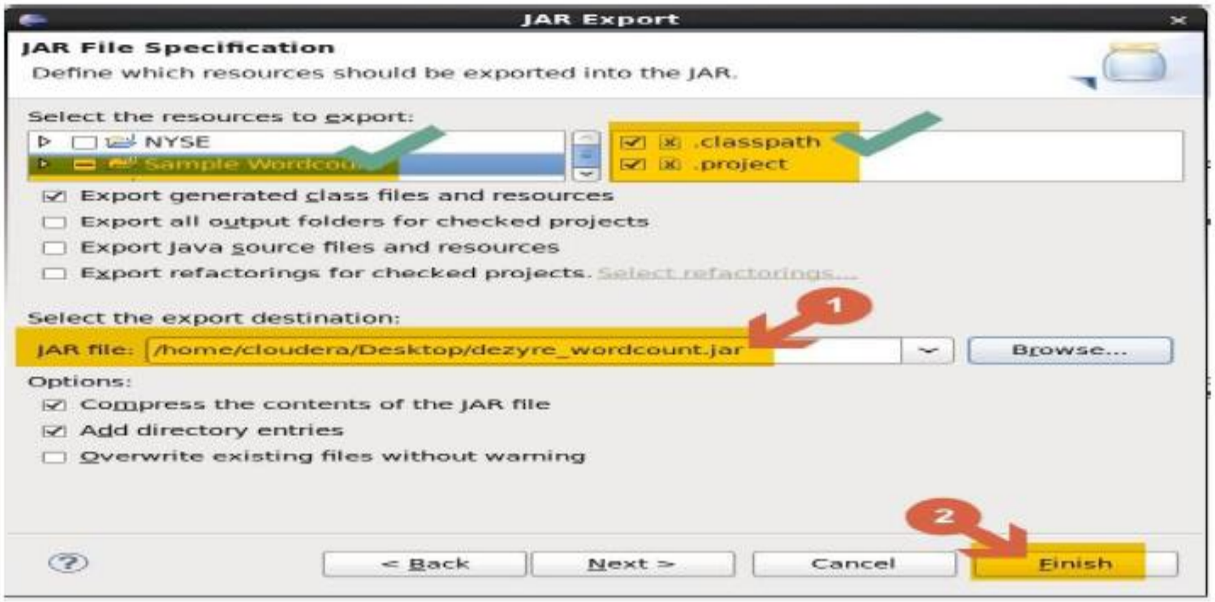
☑ Export generated class files and resources
□ Export all output folders for checked projects
□ Export java source files and resources
□ Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: /home/cloudera/Desktop/NYSE_NEU.jar

Options:

☑ Compress the contents of the JAR file
☑ Add directory entries
□ Overwrite existing files without warning



JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- ▷ □ NYSE
- ▷ ▬ Sample Wordcou...

☑ ☒ .classpath
☑ ☒ .project

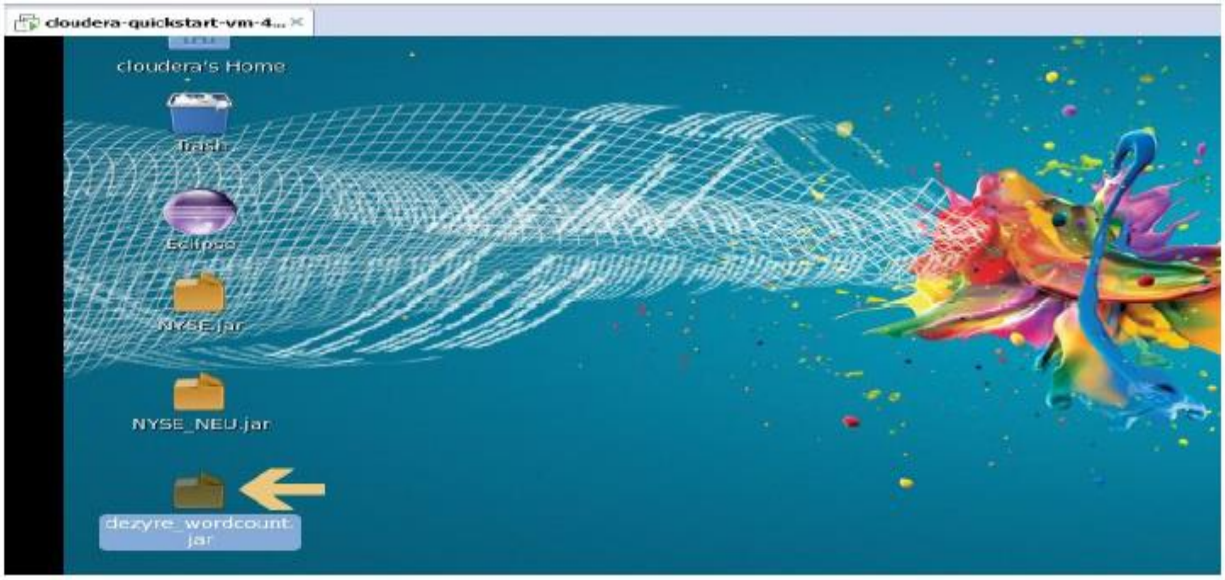
☑ Export generated class files and resources
□ Export all output folders for checked projects
□ Export java source files and resources
□ Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: /home/cloudera/Desktop/dezyre_wordcount.jar

Options:

☑ Compress the contents of the JAR file
☑ Add directory entries
□ Overwrite existing files without warning



Execute the Hadoop MapReduce WordCount program

Command1:

JPS stands for Java Virtual Machine Process Status Tool. **JPS** is a **command** is used to check all the Hadoop daemons like NameNode, DataNode, ResourceManager, NodeManager etc. JPS command displays all java based processes for a particular user.

```
root@cse-OptiPlex-3046:/home/cse#jps
8594 Jps
6980 DataNode
7190 SecondaryNameNode
7702 NodeManager
6809 NameNode
7354 ResourceManager
```

Command2:

Once all resources are running.Then Create a input file using vi command

Vi is a Screen editorin linux.vi stands for **visual instrument**. It is a widely-used default text editor for Unix-based systems and is shipped with vitually all versions of Unix.

```
root@cse-OptiPlex-3046:/home/cse# vi word.txt
```

1. To enter **vi**, type: **vi** filename
2. To enter **insert** mode, type: **i**.
3. Type in the text: **bus car train train car train bus bus**
4. To leave **insert** mode and return to **command**mode, press: **<Esc>**
5. In **command** mode, save changes and exit **vi** by typing: **:wq**<Return> You areback at the Unix prompt.

Command3:

Create a directory in hdfs

```
root@cse-OptiPlex-3046:/home/cse#hdfs dfs -mkdir /wordcountinput
```

Command4:

Upload single source, or multiple sources from local file system to the destination filesystem.

```
root@cse-OptiPlex-3046:/home/cse# hdfs dfs -put '/home/cse/word.txt' /wordcountinput
```

Command5:

http://localhost:50070/dfshealth.html#tab-overview

Click on Utilities and then browse file system.

Command6:

Yarn commands are invoked by the bin/yarn script.Run a jar file. Users can bundle theirYarn code in a jar file and execute it using this command.Run the WordCount applicationfrom the JAR file, passing the paths to the **input and output directories** in HDFS.

```
root@cse-OptiPlex-3046:/#yarn jar wordcount.jar wordcount.WordCount wordcountinput/word.txt /wordcountoutput
```

Command7:

```
root@cse-OptiPlex-3046:/# hdfs dfs -ls /wordcountoutput
Found 2 items
-rw-r--r-- 1 cse supergroup 0 2018-11-01 15:30 /wordcountoutput/_SUCCESS
-rw-r--r-- 1 cse supergroup 20 2018-11-01 15:30 /wordcountoutput/part-r-00000
```

Command8:

```
root@cse-OptiPlex-3046:/# hdfs dfs -cat wordcountoutput/part-r-00000
bus 3
car 2
train 3
```

Experiment # 5

Write a map-reduce program for finding maximum temperature in weather dataset.

Objective :: ***Temperature program in map reduce environment***

PROGRAM MODULE:

Mapper Program(Mapper.java)

```
package temperature;
import java.io.IOException;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
public class Mapper 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 Program(Reducer.java)

```
package temperature;
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
public class Reducer 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; //for minimum int max_temp = 100;

while (values.hasNext())
{
int current=values.next().get();
if ( max_temp < current) // for minimum if ( max_temp > current)
max_temp = current;
}
output.collect(key, new IntWritable(max_temp/10));
} }
```

Driver Program(Driver.java)

```
package temperature;
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 Driver extends Configured implements Tool{
public int run(String[] args) throws Exception
{
JobConf conf = new JobConf(getConf(), HighestDriver.class);
conf.setJobName("Driver");
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(IntWritable.class);
conf.setMapperClass(Mapper.class);
conf.setReducerClass(Reducer.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);
}
}
```

Execution Commands:

```
root@cse-OptiPlex-3046:/#hdfs dfs -mkdir /tempinput
root@cse-OptiPlex-3046:/#hdfs dfs -cat '/home/cse/1901.txt' /tempinput
root@cse-OptiPlex-3046:/#yarn jar temp.jar temperature.FileJoinerDriver
/tempinput/1901.txt /temoutput
root@cse-OptiPlex-3046:/#hdfs dfs -ls /tempoutput
Found 2 items
-rw-r--r-- 1 cse supergroup0 2018-11-14 15:14 /joinoutput/_SUCCESS
-rw-r--r-- 1 cse supergroup84 2018-11-14 15:14 /joinoutput/part-r-00000
root@cse-OptiPlex-3046:/#hdfs dfs -cat /tempoutput/part-r-00000
1901 23.
```

Experiment # 6

Write a map-reduce program for join of two records.

Objective :: Two records joining in Map reduce environment

PROGRAM MODULE:

Mapper Program(FileJoinerMapper.java)

```
package join;
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class FileJoinerMapper extends Mapper<LongWritable, Text, Text, Text> {
public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
String[] tokens = value.toString().split(",");
if (null != tokens && tokens.length == 2) {
context.write(new Text(tokens[0]), new Text(tokens[1]));
} } }
```

Reducer Program(FileJoinerReducer.java)

```
package join;
import java.io.IOException;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class FileJoinerReducer extends Reducer<Text, Text, NullWritable, Text> {
public void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
StringBuffer rec = new StringBuffer(key.toString()).append(",");
int count = 0;
for (Text val : values) {
rec.append(val.toString());
if (count < 1) {
```

```
rec.append(",");
} count++;
}
context.write(NullWritable.get(), new Text(rec.toString()));
} }
```

Driver Program(FileJoinerDriver.java)

```
package join;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class FileJoinerDriver {
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
String[] otherArgs = new GenericOptionsParser(conf,
args).getRemainingArgs();
if (otherArgs.length != 3) {
System.err.println("Usage: Join <in-1><in-2><out>");
System.exit(2);
}
Job job = Job.getInstance(conf, "File Joiner");
job.setJarByClass(FileJoinerDriver.class);
job.setMapperClass(FileJoinerMapper.class);
job.setReducerClass(FileJoinerReducer.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(Text.class);
job.setOutputKeyClass(NullWritable.class);
job.setOutputValueClass(Text.class);
FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
FileInputFormat.addInputPath(job, new Path(otherArgs[1]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[2]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
} }
```

Data sets:

empname.txt

101,Gaurav
102,Rohit
103,Karishma
104,Darshan
105,Divya
27

empdept.txt

101,Sales
102,Research
103,NMG
104,Admin
105,HR

Execution Commands:

```
root@cse-OptiPlex-3046:/#hdfs dfs -mkdir /joininput
root@cse-OptiPlex-3046:/#hdfs dfs -cat ‘/home/cse/empname.txt’ /joininput
root@cse-OptiPlex-3046:/#hdfs dfs -cat ‘/home/cse/empdept.txt’ /joininput
root@cse-OptiPlex-3046:/#yarn jar join.jar join.FileJoinerDriver
/joininput/empdept.txt /joininput/empname.txt /joinoutput
root@cse-OptiPlex-3046:/#hdfs dfs -ls /joinoutput
Found 2 items
-rw-r--r-- 1 cse supergroup 0 2018-11-14 15:14 /joinoutput/_SUCCESS
-rw-r--r-- 1 cse supergroup 84 2018-11-14 15:14 /joinoutput/part-r-00000
root@cse-OptiPlex-3046:/#hdfs dfs -cat /joinoutput/part-r-00000
101,Sales,Gaurav
102,Rohit,Research
103,NMG,Karishma
104,Darshan,Admin
105,HR,Divya
```

Experiment # 7

Write a map reduce program to find duplicate record in csv file.

Objective: Duplicate record finding in Map reduce Environment from csv file.

PROGRAM MODULE:

Mapper:

```
package duplicate;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class DuplicateValueMapper
extends Mapper<LongWritable, Text, Text, IntWritable>{
private static final IntWritable one = new IntWritable(1);
@Override
protected void map(LongWritable key, Text value,
Mapper<LongWritable, Text, Text, IntWritable>.Context context)
throws IOException, InterruptedException {
// TODO Auto-generated method stub
//Skipping the header of the input
if (key.get() == 0 && value.toString().contains("first_name")) {
return;
}
else {
String values[] = value.toString().split(",");
context.write(new Text(values[1]), one); //Writing first_name value as a key
}
}
}
```

Reducer:

```
package duplicate;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import com.google.common.collect.Iterables;
/*
* This reducer will get mapper data as input and return only key that is duplicate
value.
*
*/
public class DuplicateValueReducer

extends Reducer<Text, IntWritable, Text, NullWritable>{
@Override
protected void reduce(Text key, Iterable<IntWritable> values,
Reducer<Text, IntWritable, Text, NullWritable>.Context context)
throws IOException, InterruptedException {
// TODO Auto-generated method stub
//Check if the key has duplicate value
if (Iterables.size(values) > 1) {
context.write(key, NullWritable.get());
} } }
```

Driver:

```
package duplicates;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
import duplicate1.DuplicateValueMapper;
import duplicate1.DuplicateValueReducer;
public class DuplicateValueDriver
extends Configured implements Tool{
public int run(String[] arg0) throws Exception {
```

```
// TODO Auto-generated method stub
@SuppressWarnings("deprecation")
Job job = new Job(getConf(), "Duplicate value");
job.setJarByClass(getClass());
job.setMapperClass(DuplicateValueMapper.class);
job.setReducerClass(DuplicateValueReducer.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(arg0[0]));
FileOutputFormat.setOutputPath(job, new Path(arg0[1]));

return job.waitForCompletion(true) ? 0 : 1;
}
public static void main(String[] args) throws Exception {
int jobStatus = ToolRunner.run(new DuplicateValueDriver(), args);
System.out.println(jobStatus);
}
}
```

Execution Commands:

```
cse@cse-OptiPlex-3046:~$ hdfs dfs -mkdir /duplicate
cse@cse-OptiPlex-3046:~$ hdfs dfs -put '/home/cse/duplicate.csv' /duplicate
cse@cse-OptiPlex-3046:~$ yarn jar duplicate.jar
duplicate1.DuplicateValueDriver /duplicate/duplicate.csv /duplicateout
cse@cse-OptiPlex-3046:~$ hdfs dfs -cat /duplicateout/part-r-00000
Celie
Hercule
```

Experiment # 8

Write a map reduce program to find patent citations in patent dataset

Objective :: **Patent citation program in Map reduce environment**

PROGRAM MODULE:

```
package com.citation;
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
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.input.KeyValueTextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
public class PatentCitation {
public static class PatentCitationMapper extends Mapper<Text, Text, Text, Text> {
public void map(Text key, Text value, Context context) throws IOException,
InterruptedException {
String[] citation = key.toString().split( "," );
Text cited = new Text(citation[1]);
Text citing = new Text(citation[0]);
context.write(cited, citing);
} }
public static class PatentCitationReducer extends Reducer<Text, Text, Text, Text> {
@Override
protected void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
String csv = "";
for (Text value : values) {
```

```
if (csv.length() > 0) {
csv += ",";
} csv += value.toString();
} context.write(key, new Text(csv));
}  }
public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "Hadoop Patent Citation" );
job.setJarByClass(PatentCitation.class );
job.setMapperClass(PatentCitationMapper.class);
job.setCombinerClass(PatentCitationReducer.class);
job.setReducerClass(PatentCitationReducer.class);
job.setInputFormatClass(KeyValueTextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(Text.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1 );
} }
```

Execution Commands:

```
root@cse-OptiPlex-3046:/#hdfs dfs -mkdir /citationinp
root@cse-OptiPlex-3046:/#hdfs dfs -cat ‘/home/cse/cite75_99.txt’
/citationinp
root@cse-OptiPlex-3046:/#yarn jar citation.jar com.citation.PatentCitation
/citationinp/empdept.txt /citationout
root@cse-OptiPlex-3046:/#hdfs dfs -ls /citationout
Found 2 items
-rw-r--r-- 1 cse supergroup 0 2018-11-14 15:14 /citationout/_SUCCESS
-rw-r--r-- 1 cse supergroup 84 2018-11-14 15:14 /citationout/part-r-00000
root@cse-OptiPlex-3046:/#hdfs dfs -cat /citationout/part-r-00000
955948 5794647
955954 5288283,5445585
955955 4001940,4768950
955957 5203827
955959 4429622
955970 3969088,4184456
```

Experiment # 9

Write a Map reduce program for total retail collection.(retail dataset)

Objective: Finding total retail collection in Map reduce environment.

PROGRAM MODULE:

Mapper.java

```
package retailtotal;
import java.io.IOException;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class RetailDataAnalysisMapper extends Mapper<LongWritable, Text, Text,
FloatWritable> {
private FloatWritable percentVal = new FloatWritable();
private Text moKey = new Text();
public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
//Date Time City Product-Category Sale-Vale Payment-Mode
//2012-01-01 09:00 San Jose Men's Clothing214.05 Amex
try {
String valueTokens[] = value.toString().split("\t");
float saleValue ;
if (valueTokens.length > 0 && valueTokens.length == 6) {
moKey.set("All Stores ");
saleValue = Float.parseFloat(valueTokens[4]);
percentVal.set(saleValue);
context.write(moKey, percentVal);
} } catch(Exception e) {
e.printStackTrace(); } } }
```

Reducer.java

```
package retailtotal;
import java.io.IOException;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class RetailDataAnalysisReducer extends Reducer<Text, FloatWritable, Text,
FloatWritable> {
private FloatWritable result = new FloatWritable();
public void reduce(Text key, Iterable<FloatWritable> values, Context context)
throws IOException, InterruptedException {
float sum = 0.0f;
int count = 0;
for (FloatWritable val : values) {
count += 1;
sum += val.get();
}
result.set(sum);
String reduceKey = "Number of sales " + String.valueOf(count) + ", Sales
Value : ";
context.write(new Text(reduceKey), result);
}
}
```

Driver:

```
package retailtotal;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class RetailDataAnalysis {
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
String[] otherArgs = new GenericOptionsParser(conf,
args).getRemainingArgs();
if (otherArgs.length != 2) {
System.err.println("Usage: Number Sum <in><out>");
System.exit(2);
}
Job job = Job.getInstance(conf, "Retail Data All Store Analysis");
job.setJarByClass(RetailDataAnalysis.class);
job.setMapperClass(RetailDataAnalysisMapper.class);
job.setReducerClass(RetailDataAnalysisReducer.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(FloatWritable.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(FloatWritable.class);
FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

Execution Commands:

```
cse@cse-OptiPlex-3046:~$ hdfs dfs -mkdir /retail
cse@cse-OptiPlex-3046:~$ hdfs dfs -put '/home/cse/Retail.txt' /retail
cse@cse-OptiPlex-3046:~$ yarn jar retailtotal.jar retailtotal.RetailDataAnalysis
/retail/Retail.txt /retailout2
cse@cse-OptiPlex-3046:~$ hdfs dfs -cat /retailout2/part-r-00000
Number of sales 200, Sales Value : 49585.363
```

Experiment # 10

Write a Map reduce program for store wise collection. (retail dataset)

Objective: ***Perform the Map reduce program on retail dataset for store wise collection***

Program Module:

Mapper:

```
package retailstore;
import java.io.IOException;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class RetailDataAnalysisMapper extends Mapper<LongWritable, Text, Text,
FloatWritable> {
private FloatWritable percentVal = new FloatWritable();
private Text moKey = new Text();
public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
// Date Time City Product-Category Sale-Vale Payment-Mode
// 2012-01-01 09:00 San Jose Men's Clothing 214.05 Amex
try {
String valueTokens[] = value.toString().split("\t");
String date, store;
float saleValue;
if (valueTokens.length > 0 && valueTokens.length == 6) {
date = valueTokens[0];
store = valueTokens[2];
moKey.set(date + "\t" + store);
saleValue = Float.parseFloat(valueTokens[4]);
percentVal.set(saleValue);
context.write(moKey, percentVal);
}
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

Reducer:

```
package retailstore;
import java.io.IOException;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class RetailDataAnalysisReducer extends Reducer<Text, FloatWritable, Text,
FloatWritable> {
private FloatWritable result = new FloatWritable();
public void reduce(Text key, Iterable<FloatWritable> values, Context context)
throws IOException, InterruptedException {
float sum = 0.0f;
for (FloatWritable val : values) {
sum += val.get();
}
result.set(sum);
context.write(key, result);
} }
```

Driver:

```
package retailstore;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class RetailDataAnalysis {
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
String[] otherArgs = new GenericOptionsParser(conf,
args).getRemainingArgs();
if (otherArgs.length != 2) {
System.err.println("Usage: Number Sum <in><out>");
System.exit(2);
}
Job job = Job.getInstance(conf, "Retail Data Store Analysis");
job.setJarByClass(RetailDataAnalysis.class);
job.setMapperClass(RetailDataAnalysisMapper.class);
job.setReducerClass(RetailDataAnalysisReducer.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(FloatWritable.class);

job.setOutputKeyClass(Text.class);
job.setOutputValueClass(FloatWritable.class);
FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

Execution Commands:

```
cse@cse-OptiPlex-3046:~$ hdfs dfs -mkdir /retail
cse@cse-OptiPlex-3046:~$ hdfs dfs -put '/home/cse/Retail.txt' /retail
cse@cse-OptiPlex-3046:~$ yarn jar retailstore.jar
retailstore.RetailDataAnalysis
/retail/Retail.txt /retailout1
cse@cse-OptiPlex-3046:~$ hdfs dfs -cat /retailout1/part-r-00000
2012-01-01 Albuquerque 1074.88
2012-01-01 Anaheim 114.41
2012-01-01 Anchorage 1086.22
2012-01-01 Arlington 400.08
2012-01-01 Atlanta 254.62
2012-01-01 Aurora 117.81
2012-01-01 Austin 1787.88
2012-01-01 Bakersfield 217.79
2012-01-01 Baltimore 7.98
2012-01-01 Boise 481.08997
2012-01-01 Boston 1114.54
2012-01-01 Buffalo 483.82
2012-01-01 Chandler 1648.7699
2012-01-01 Charlotte 440.11
2012-01-01 Chesapeake 676.35
2012-01-01 Chicago 146.15
```

```
2012-01-01 Cincinnati 323.37997
2012-01-01 Cleveland 427.43
2012-01-01 Columbus 392.5
2012-01-01 Corpus Christi 25.38
2012-01-01 Dallas 273.49
2012-01-01 Denver 413.21002
2012-01-01 Detroit 134.89
2012-01-01 Durham 980.32007
2012-01-01 El Paso 103.01
2012-01-01 Fort Wayne 370.55
2012-01-01 Fort Worth 1128.1399
```

Experiment # 11

Write a Map reduce program for product wise collection.(retail dataset)

Objective:: ***Perform the Map reduce program on retail dataset for product wise collection***

PROGRAM MODULE:

Mapper:

```
package retailproduct;
import java.io.IOException;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class RetailDataAnalysisMapper extends Mapper<LongWritable, Text, Text,
FloatWritable> {
private FloatWritable percentVal = new FloatWritable();
private Text moKey = new Text();
public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
//Date Time City Product-Category Sale-Vale Payment-Mode
//2012-01-01 09:00 San Jose Men's Clothing214.05 Amex
try {
String valueTokens[] = value.toString().split("\t");
String date = valueTokens[0];
String productCat = valueTokens[3];
float saleValue ;
if (valueTokens.length > 0 && valueTokens.length == 6) {
moKey.set(date + "\t" + productCat);
saleValue = Float.parseFloat(valueTokens[4]);
percentVal.set(saleValue);
context.write(moKey, percentVal);
}
} catch(Exception e) {
e.printStackTrace();}}}
```

Reducer:

```
package retailproduct;
import java.io.IOException;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class RetailDataAnalysisReducer extends Reducer<Text, FloatWritable, Text,
FloatWritable> {
private FloatWritable result = new FloatWritable();
public void reduce(Text key, Iterable<FloatWritable> values, Context context)
throws IOException, InterruptedException {
float sum = 0.0f;
for (FloatWritable val : values) {
sum += val.get();
}
result.set(sum);
context.write(key, result);
}
}
```

Driver:

```
package retailproduct;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class RetailDataAnalysis {
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
String[] otherArgs = new GenericOptionsParser(conf,
args).getRemainingArgs();
if (otherArgs.length != 2) {
System.err.println("Usage: Number Sum <in><out>");
System.exit(2);
}
Job job = Job.getInstance(conf, "Retail Data Product Analysis");
job.setJarByClass(RetailDataAnalysis.class);
job.setMapperClass(RetailDataAnalysisMapper.class);
job.setReducerClass(RetailDataAnalysisReducer.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(FloatWritable.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(FloatWritable.class);
FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

Execution Commands:

```
        cse@cse-OptiPlex-3046:~$ hdfs dfs -mkdir /retail
        cse@cse-OptiPlex-3046:~$ hdfs dfs -put '/home/cse/Retail.txt' /retail
        cse@cse-OptiPlex-3046:~$ yarn jar retailproduct.jar
        retailproduct.RetailDataAnalysis /retail/Retail.txt /retailout
        cse@cse-OptiPlex-3046:~$ hdfs dfs -cat /retailout/part-r-00000
        2012-01-01 Baby 2034.23
        2012-01-01 Books 3492.8
        2012-01-01 CDs 2644.5098
        2012-01-01 Cameras 2591.27
        2012-01-01 Children's Clothing 2778.21
        2012-01-01 Computers 2102.66
        2012-01-01 Consumer Electronics 2963.59
        2012-01-01 Crafts 3258.0898
        2012-01-01 DVDs 2831.0
        2012-01-01 Garden 1882.25
        2012-01-01 Health and Beauty 2467.3198
        2012-01-01 Men's Clothing 4030.89
        2012-01-01 Music 2396.4
        2012-01-01 Pet Supplies 2660.83
        2012-01-01 Sporting Goods 1952.89
        2012-01-01 Toys 3188.18
        2012-01-01 Video Games 2573.3801
        2012-01-01 Women's Clothing 3736.87
```

Department of Computer Science and Engineering

SRKR Engineering College (A), Bhimavaram, India