S.R.K.R ENGINEERING COLLEGE(A)

(Affiliated to JNTU KAKINADA)

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

CHINNAMIRAM::BHIMAVARAM - 534204

(2021-2022)

BIG DATA ANALYTICS LAB

LABORATORY RECORD

Name :

Reg No :

Year : IV

Sem : I

Branch : CSE

INDEX

Signature of Lab Incharge

1. Setting up Hadoop on a standalone machine.

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 component as they could communicate across network sockets, effectively producing a fully functioning and optimized mini-cluster on a single host.

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 the user‟s home directory and add following parameters.

Command :nano .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 script to run Hadoop like Java_home path, etc. Edit configuration file hadoop-env.sh(located in HADOOP_HOME/etc/hadoop) and set JAVA_HOME.

Command: hadoop–env.sh

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

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 are common to HDFS & MapReduce. Edit configuration file core-site.xml (located inHADOOP_HOME/etc/hadoop) and add following entries.

Command :nano 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 factor and block size of HDFS. Edit configuration file hdfs-site.xml (located inHADOOP_HOME/etc/hadoop) and add following entries.

Command: nano 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 Reducer process, CPU cores available for a process, etc.In some cases, mapred-site.xml file is not available. So, we have to create the mapred-site.xml file using mapred-site.xml template. Edit configuration file mapred-site.xml (located in HADOOP_HOME/etc/hadoop) and add following entries.

Command: nano mapred-site.xml

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

4.6 Edit yarn-site.xml

yarn-site.xml contains 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:nano 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

2. Setting up hadoop in a distributed cluster environment.

1.Prerequisites for Hadoop Multi Node Cluster Setup

1.1 Add Entries in hosts file

Edit hosts file and add entries of both master and slaves

sudo nano /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

1.2 Install Java 8 (Recommended Oracle Java)

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

Command: sudo apt update

Command: sudo apt install openjdk-8-jdk

Command: export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

Command: echo $JAVA_HOME

1.3 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

2. Install Hadoop

2.1 Download Hadoop

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

2.2 .Untar Tar ball

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

3. Hadoop multi-node cluster setup Configuration

3.1 Edit .bashrc

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

Command : nano .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

3.2 Edit hadoop-env.sh

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

Command : nano hadoop–env.sh

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

3.3 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 are common to HDFS & MapReduce.

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

Command : nano 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>
```

3.4 Edit hdfs-site.xml

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

Command : nano hdfs-site.xml

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

3.5 Edit mapred-site.xml

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

Command : nano mapred-site.xml

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

3.6 Edit yarn-site.xml

yarn-site.xml contains configuration settings of ResourceManager and NodeManager 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 : nano 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 add following entries:

slave01

slave02

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

Install Hadoop On Slaves

4. 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)

5. Copy configured setups from master to all the slaves

5.1. Create tarball of configured setup

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

(NOTE: Run this command on Master)

5.2. 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)

5.3. 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.

6. Start the Hadoop Cluster

Let us now learn how to start a Hadoop cluster?

6.1. Format the name node

Command : bin/hdfs namenode -format

(Note: Run this command on Master)

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

6.2. Start HDFS Services

Command : sbin/start-dfs.sh

(Note: Run this command on Master)

6.3. Start YARN Services

Command :sbin/start-yarn.sh

(Note: Run this command on Master)

6.4. Check for Hadoop services

6.4.1. Check daemons on Master

Command : jps

NameNode

ResourceManager

6.4.2. Check daemons on Slaves

Command : jps

DataNode

NodeManager

7. Stop The Hadoop Cluster

Let us now see how to stop the Hadoop cluster? (Note: Run this command on Master)

7.1. Stop HDFS and YARN Services

Command : sbin/stop-yarn.sh

Command : sbin/stop-dfs.sh

3. Write any Five Hadoop commands.

1. **Create a directory**:

   hadoop fs -mkdir /foldername

   example:hadoop fs -mkdir /input

2. **Create a text file**:

   gedit filename.extension

   example:gedit sample.txt

3. **Upload into HDFS** :

   hadoop fs -put „/local file path/filename.extension /hdfs folder name

   example:hadoop fs -put '/home/cse/sample.txt' /input

4. **Download HDFS to local**:

   hadoop fs -get '/hdfs path of a file/file.extension'/local folder name

   example:hadoop fs -get '/inp/sample.txt' home/cse

5. **To display the content of a file** :

   hadoop fs -cat /path of a file/filename.extension

   example:hadoop fs -cat input/sample.txt

6. **To display first few lines of a file**:

   hadoop fs -cat /path of a file/filename.extension |head-number

   example:hadoop fs -cat /inp/sampe.txt |head -5

7. **To display last few lines of a file:**

   hadoop fs -cat /path of a file /filename.extension |tail -number

   example:hadoop fs -cat /inp/sampe.txt |tail -10

8. **Remove a file or directory:**

   hadoop fs -rm /path of a file or directory

   example:hadoop fs -rm np/sample.txt

9. **Copy a file from source to destination:**

   hadoop fs -cp sourcefile.extension destinationfolder

   example: hadoop fs – cp home/cse/sample.txt home/desktop

4. Write a Map reduce program for word count.(with Combiner).

DRIVER CLASS:

```
package sriram;
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(TokenizerMapper.class);
		job.setCombinerClass(IntSumReducer.class);
		job.setReducerClass(IntSumReducer.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);
	}
}
```

MAPPER CLASS:

```
package sriram;
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 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);
		}
	}
}
```

REDUCER CLASS:

```
package sriram;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public 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();
		}
		result.set(sum);
```

```
        context.write(key, result);}
```

OUTPUT:

```
root@cse-OptiPlex-3046:/home/cse#hadoop fs -mkdir /wordcountinput
root@cse-OptiPlex-3046:/home/cse# hadoop fs -put '/home/cse/word.txt'
/wordcountinput
root@cse-OptiPlex-3046:/#hadoop jar wordcount.jar wordcount.WordCount
wordcountinput/word.txt /wordcountoutput
root@cse-OptiPlex-3046:/# hadoop fs -ls /wordcountoutput
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:/# hadoop fs -cat wordcountoutput/part-r-00000
a 4
b 5
c 6
d 7
```

5. Write a map reduce program to find minimum temperature.

DRIVER CLASS:

```
package sriram;
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;

public class MinTemperature {

        public static void main(String[] args) throws Exception {
        if (args.length != 2) {
                System.err.println("Usage: MinTemperature <input path> <output path>");
                System.exit(-1);
        }
        Job job = new Job();
        job.setJarByClass(MinTemperature.class);
        job.setJobName("Min temperature");
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.setMapperClass(MinTemperatureMapper.class);
        job.setReducerClass(MinTemperatureReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
        }
}
```

MAPPER CLASS:

```
package sriram;
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;
```

```
public class MinTemperatureMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

	private static final int MISSING = 9999;
	@Override
	public void map(LongWritable key, Text value, Context context) throws IOException,
	InterruptedException {
	String line = value.toString();
	String year = line.substring(15, 19);
	int airTemperature;
	if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs
		airTemperature = Integer.parseInt(line.substring(88, 92));
	} else {
		airTemperature = Integer.parseInt(line.substring(87, 92));
	}
	String quality = line.substring(92, 93);
	If (airTemperature != MISSING && quality.matches("[01459]")) {
		context.write(new Text(year), new IntWritable(airTemperature));
	}
}}
```

REDUCER CLASS:

```
package sriram;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MinTemperatureReducer extends Reducer<Text, IntWritable, Text, IntWritable>
{
	public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
	IOException, InterruptedException {
	int minValue = Integer.MAX_VALUE;
	for (IntWritable value : values) {
		minValue = Math.min(minValue, value.get());
	}
	context.write(key, new IntWritable(minValue));
	}
```

}

OUTPUT:

root@cse-OptiPlex-3046:hadoop fs -mkdir /tempinput

root@cse-OptiPlex-3046:hadoop fs -cat ‘/home/cse/1901.txt’ /tempinput

root@cse-OptiPlex-3046:hadoop jar temp.jar sriram.MinTemperature /tempinput/1901.txt /temoutput

root@cse-OptiPlex-3046:hadoop fs -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:hadoop fs -cat /tempoutput/part-r-00000

1901 23.

6. Write a map reduce program to find maximum temperature.

DRIVER CLASS:

```
package sriram;
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;

public class MaxTemperature {

        public static void main(String[] args) throws Exception {
        if (args.length != 2) {
                System.err.println("Usage: MaxTemperature <input path> <output path>");
                System.exit(-1);
        }
        Job job = new Job();
        job.setJarByClass(MaxTemperature.class);
        job.setJobName("Max temperature");
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.setMapperClass(MaxTemperatureMapper.class);
        job.setReducerClass(MaxTemperatureReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
        }
}
```

REDUCER CLASS:

```
package sriram;
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;

public class MaxTemperatureMapper extends Mapper<LongWritable, Text, Text,
```

```
IntWritable> {

        private static final int MISSING = 9999;
        @Override
        public void map(LongWritable key, Text value, Context context) throws IOException,
        InterruptedException {
        String line = value.toString();
        String year = line.substring(15, 19);
        int airTemperature;
        if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs
                airTemperature = Integer.parseInt(line.substring(88, 92));
        } else {
                airTemperature = Integer.parseInt(line.substring(87, 92));
        }
        String quality = line.substring(92, 93);
        if (airTemperature != MISSING && quality.matches("[01459]")) {
                context.write(new Text(year), new IntWritable(airTemperature));
        }
        }
}
```

MAPPER CLASS:

```
package sriram;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MaxTemperatureReducer extends Reducer<Text, IntWritable, Text, IntWritable>
{
        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
        IOException, InterruptedException {
        int maxValue = Integer.MIN_VALUE;
        for (IntWritable value : values) {
                maxValue = Math.max(maxValue, value.get());
        }
        context.write(key, new IntWritable(maxValue));
        }
```

}

OUTPUT:

root@cse-OptiPlex-3046:hadoop fs -mkdir /tempinput
root@cse-OptiPlex-3046:hadoop fs -cat ‘/home/cse/1901.txt’ /tempinput
root@cse-OptiPlex-3046:hadoop jar temp.jar sriram.MaxTemperature
/tempinput/1901.txt /output
root@cse-OptiPlex-3046:hadoop fs -ls /output
Found 2 items
-rw-r--r-- 1 cse supergroup0 2018-11-14 15:14 /output/_SUCCESS
-rw-r--r-- 1 cse supergroup84 2018-11-14 15:14 /output/part-r-00000
root@cse-OptiPlex-3046:hadoop fs -cat /tempoutput/part-r-00000
1901 256

7. Write a map reduce program to find the store wise collection.

DRIVER CLASS:

```
package sriram;
import org.apache.hadoop.conf.Configuration;
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
import  org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.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);
	}

}
```

MAPPER CLASS:

```
package sriram;
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{
                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 CLASS:

```
package sriram;
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);
	}
}
```

OUTPUT:

```
root@cse-OptiPlex-3046:hadoop fs -mkdir /input
root@cse-OptiPlex-3046:hadoop fs -cat '/home/cse/Retail.txt' /input
root@cse-OptiPlex-3046:hadoop jar retail.jar sriram.RetailDataAnalysis
/input/Retail.txt /output
root@cse-OptiPlex-3046:hadoop fs -ls /output
Found 2 items
-rw-r--r-- 1 cse supergroup0 2018-11-14 15:14 /output/_SUCCESS
-rw-r--r-- 1 cse supergroup84 2018-11-14 15:14 /output/part-r-00000
root@cse-OptiPlex-3046:hadoop fs -cat /tempoutput/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
```

8. Write a map reduce program to find the total retail collection across all the retail stores.

DRIVER CLASS:

```
package sriram;
import org.apache.hadoop.conf.Configuration;
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
import  org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.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("Usgae: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);
	}
}
```

MAPPER CLASS:

```
package sriram;
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{
                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 CLASS:

```
package sriram;
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(key,result);
        }
}
```

OUTPUT:

```
root@cse-OptiPlex-3046:hadoop fs -mkdir /input
root@cse-OptiPlex-3046:hadoop fs -cat '/home/cse/Retail.txt' /input
root@cse-OptiPlex-3046:hadoop jar retail.jar sriram.RetailDataAnalysis
/input/Retail.txt /output
root@cse-OptiPlex-3046:hadoop fs -ls /output
Found 2 items
-rw-r--r-- 1 cse supergroup0 2018-11-14 15:14 /output/_SUCCESS
-rw-r--r-- 1 cse supergroup84 2018-11-14 15:14 /output/part-r-00000
root@cse-OptiPlex-3046:hadoop fs -cat /output/part-r-00000
Number of sales 200, Sales Value : 49585.363
```

9. Write a map reduce program to find the total product wise sales of all the retail stores.

DRIVER CLASS:

```
package sriram;
import org.apache.hadoop.conf.Configuration;
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
import  org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.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 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);
        }
}
```

MAPPER CLASS:

```
package sriram;
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{
                try {
                        String valueTokens[]=value.toString().split("\t");
                        float saleValue;
                        String date=valueTokens[0];
                        String productCat=valueTokens[3];
                        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 CLASS:

```
package sriram;
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);
        }
}
```

OUTPUT:

root@cse-OptiPlex-3046:hadoop fs -mkdir /input

root@cse-OptiPlex-3046:hadoop fs -cat '/home/cse/Retail.txt' /input

root@cse-OptiPlex-3046:hadoop jar retail.jar sriram.RetailDataAnalysis
/input/Retail.txt /output

root@cse-OptiPlex-3046:hadoop fs -ls /output

Found 2 items

-rw-r--r-- 1 cse supergroup0 2018-11-14 15:14 /output/_SUCCESS

-rw-r--r-- 1 cse supergroup84 2018-11-14 15:14 /output/part-r-00000

root@cse-OptiPlex-3046:hadoop fs -cat /output/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    3736.8
```

10. Write a map reduce program to find Duplicate Records in a CSV file.

DRIVER CLASS:

```
package sriram;
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 sriram.DuplicateValueMapper;
import sriram.DuplicateValueReducer;

public class DuplicateValueDriver extends Configured implements Tool{
        public int run(String[] arg0) throws Exception {
                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);
        }
}
```

MAPPER CLASS:

```
package sriram;
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 CLASS:

```
package sriram;
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;

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 {
        if (Iterables.size(values) > 1) {
                context.write(key, NullWritable.get());
        }
        }
}
```

OUTPUT:

root@cse-OptiPlex-3046:hadoop fs -mkdir /input
root@cse-OptiPlex-3046:hadoop fs -cat '/home/cse/Duplicate.csv' /input
root@cse-OptiPlex-3046:hadoop jar Dup.jar sriram.DuplicateValueDriver
/input/Duplicate.csv /output
root@cse-OptiPlex-3046:hadoop fs -ls /output
Found 2 items
-rw-r--r-- 1 cse supergroup0 2018-11-14 15:14 /output/_SUCCESS
-rw-r--r-- 1 cse supergroup84 2018-11-14 15:14 /output/part-r-00000
root@cse-OptiPlex-3046:hadoop fs -cat /output/part-r-00000
Celie
Hercule

11. Write a map reduce program to find Maximum CTC of employees in each Department.

DRIVER CLASS:

```
package sriram;
import java.io.IOException;
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 EmployeeCtcDriver {

public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException {
        Configuration conf = new Configuration();
        String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
        if (otherArgs.length != 2) {
                System.err.println("Usage: Employee CTC Analysis <input> <output>");
                System.exit(2);
        }
        Job job = Job.getInstance(conf, "Employee CTC Analysis");
        job.setJobName("Customer Partitioner");
        job.setJarByClass(EmployeeCtcDriver.class);
        job.setMapperClass(EmployeeCtcMapper.class);
        job.setReducerClass(EmployeeCtcReducer.class);
        job.setNumReduceTasks(2);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);
        job.setOutputKeyClass(NullWritable.class);
        job.setOutputValueClass(Text.class);
        FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
        FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
        }}
```

MAPPER CLASS:

```
package sriram;
import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class EmployeeCtcMapper extends Mapper<Object, Text, Text, Text> {

	public void map(Object key, Text value, Context context) throws IOException,
	InterruptedException {
		String[] tokens = value.toString().split("\t");
		String dept = tokens[4].toString();
		context.write(new Text(dept), value);
	}
}
```

REDUCER CLASS:

```
package sriram;
import java.io.IOException;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class EmployeeCtcReducer extends Reducer<Text, Text, NullWritable, Text> {
	@Override
	public void reduce(Text key, Iterable<Text> values, Context context) throws
	IOException, InterruptedException {
		int maxCtc = Integer.MIN_VALUE;
		int ctc = 0;
		String value = "";
		for (Text val : values) {
			String[] valTokens = val.toString().split("\\t");
			ctc = Integer.parseInt(valTokens[5]);
			if (ctc > maxCtc) {
				value = val.toString();
				maxCtc = ctc;
			}
		}
		context.write(NullWritable.get(), new Text(value));
	}
```

}

OUTPUT:

root@cse-OptiPlex-3046:hadoop fs -mkdir /input

root@cse-OptiPlex-3046:hadoop fs -cat '/home/cse/emp.txt' /input

root@cse-OptiPlex-3046:hadoop jar Dup.jar sriram.EmployeeCtcDriver /input/emp.txt /output

root@cse-OptiPlex-3046:hadoop fs -ls /output

Found 2 items

-rw-r--r-- 1 cse supergroup0 2018-11-14 15:14 /output/_SUCCESS

-rw-r--r-- 1 cse supergroup84 2018-11-14 15:14 /output/part-r-00000

root@cse-OptiPlex-3046:hadoop fs -cat /output/part-r-00000

13 raj 45 m cse 234

4 gg 45 m ece 105

17 wwf 56 m eee 270

14 tvp 54 f it 78

12 pvp 65 m mech 300

12. INSTALLATION OF PIG.

1.INTRODUCTION

Apache Pig is a platform or a tool that is developed originally at Facebook and is used to perform MapReduce tasks on huge datasets. It is basically used to carry out the operations on top of Hadoop . Executing Pig commands for MapReduce tasks is fairly easy to perform and easy to understand for the people having difficulty with Java coding for MapReduce. We will be looking at how to install Apache Pig in Ubuntu.

Pre-requisites

The only prerequisites for installing Apache Pig are: you should have Java and Hadoop installed.

2.INSTALL APACHE PIG IN UBUNTU

Download Apache Pig Create a new directory and download the apache pig tar.gz file in it with the below commands,

3.DOWNLOADING APACHE PIG.

If you want to download a different version of Apache Pig, you can find it here: Apache Pig Releases. Extract the Apache Pig tar file Extract the tar.gz file that you downloaded in the first step. Execute the below command,

command :tar zxvf pig-0.12.0-cdh5.3.2.tar

4.SET THE ENVIRONMENT VARIABLES.

Now we need to set the Environment Variables of Apache Pig. This process is done so that Pig can be accessed from any directory. We need to edit the .bashrc file to set the variables. Execute the below command to edit the .bashrc file

Command: nano .bashrc

Add the below lines in the file

export PIG_HOME=/home/dataflair/pig-0.12.0-cdh5.3.2
export PATH=$PATH:/home/dataflair/pig-0.12.0-cdh5.3.2/bin
export PIG_CLASSPATH=$HADOOP_HOME/conf[/php]

Save and exit the file by pressing “Ctrl X” followed by “Y” and “Enter” keys. Execute the below command to persist the changes in the .bashrc file. Pig Version We have successfully installed Apache Pig in Ubuntu.

5.VERIFYING PIG INSTALLATION

Now check the Pig version to verify the installation. Execute the below command

Command: pig -version

6. START APACHE PIG

When we start Apache Pig, it opens a grunt shell. We can start Apache Pig in 2 execution modes as below a Local Mode: In Local mode, the execution of Pig commands will be performed on the local file system. Files will be read and written from and into the local file system only rather than HDFS.

We can start Pig in Local Mode with the below command Apache Pig Local Mode

Command: pig -x local

MapReduce Mode: In this mode, the Pig commands will be executed on the files present on HDFS. The file will be read from and written into HDFS. This is the default mode of Pig. We can start Pig in MapReduce Mode with the below commands Apache Pig in MapReduce/default Mode

Command: pig -x mapreduce

13. Write a pig script to implement the FILTER operator.

```
student_details = LOAD 'home/cse/Desktop/pig_data/student_details.txt' USING PigStorage(',')
as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);
filter_data = FILTER student_details BY city == 'Chennai';
Dump filter_data;
```

INPUT:

```
001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivendram
008,Bharathi,Nambiayar,24,9848022333,Chennai
```

OUTPUT:

```
(6,Archana,Mishra,23,9848022335,Chennai)
(8,Bharathi,Nambiayar,24,9848022333,Chennai)
```

14. Write a pig script to implement FOREACH operator.

```
student_details = LOAD 'home/cse/Desktop/pig_data/student_details.txt' USING PigStorage(',')
as (id:int, firstname:chararray, lastname:chararray,age:int, phone:chararray, city:chararray);
foreach_data = FOREACH student_details GENERATE id,age,city;
Dump foreach_data;
```

INPUT:

```
001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,23,9848022335,Chennai
```

OUTPUT:

```
(1,21,Hyderabad)
(2,22,Kolkata)
(3,22,Delhi)
(4,21,Pune)
(5,23,Bhuwaneshwar)
```

15. Write a pig script to implement GROUP BY operator.

student_details = LOAD 'home/cse/Desktop/pig_data/student_details.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);

GROUP:

group_data = GROUP student_details by age;
Dump group_data;

Input:

001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddharth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanty,23,9848022336,Bhubaneswar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivandrum
008,Bharathi,Nambiayar,24,9848022333,Chennai

Output:

(21,{(4,Preethi,Agarwal,21,9848022330,Pune),(1,Rajiv,Reddy,21,9848022337,Hyderabad)})
(22,{(3,Rajesh,Khanna,22,9848022339,Delhi),(2,siddharth,Battacharya,22,984802233 8,Kolkata)})
(23,{(6,Archana,Mishra,23,9848022335,Chennai),(5,Trupthi,Mohanthy,23,9848022336, Bhuwaneshwar)})
(24,{(8,Bharathi,Nambiayar,24,9848022333,Chennai),(7,Komal,Nayak,24,9848022334, trivendram)})

GROUP MULTIPLE:

group_all = GROUP student_details All;
Dump group_all;

Input:

001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddharth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanty,23,9848022336,Bhubaneswar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivandrum

008,Bharathi,Nambiayar,24,9848022333,Chennai

Output:

(all,{(8,Bharathi,Nambiayar,24,9848022333,Chennai),(7,Komal,Nayak,24,9848022334, trivendram), (6,Archana,Mishra,23,9848022335,Chennai),(5,Tirupati,Mohanthy,23,9848022336, Bhuvaneshwar),(4,Preethi,Agarwal,21,9848022330,Pune),(3,Rajesh,Khanna,22,9848022339,Delhi), (2,siddarth,Battacharya,22,9848022338,Kolkata),(1,Rajiv,Reddy,21,9848022337,Hyderabad)})

16. Write a pig script to implement JOIN operators.

SELF JOIN:

customers = LOAD 'home/cse/Desktop/pig_data/customers.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, address:chararray, salary:int);

customers2 = LOAD 'home/cse/Desktop/pig_data/customers.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, address:chararray, salary:int);

customers3 = JOIN customers1 BY id, customers2 BY id;

Dump customers3;

INPUT:

1,Ramesh,32,Ahmedabad,2000.00

2,Khilan,25,Delhi,1500.00

3,kaushik,23,Kota,2000.00

4,Chaitali,25,Mumbai,6500.00

5,Hardik,27,Bhopal,8500.00

6,Komal,22,MP,4500.00

7,Muffy,24,Indore,10000.00

OUTPUT:

(1,Ramesh,32,Ahmedabad,2000,1,Ramesh,32,Ahmedabad,2000)

(2,Khilan,25,Delhi,1500,2,Khilan,25,Delhi,1500)

(3,kaushik,23,Kota,2000,3,kaushik,23,Kota,2000)

(4,Chaitali,25,Mumbai,6500,4,Chaitali,25,Mumbai,6500)

(5,Hardik,27,Bhopal,8500,5,Hardik,27,Bhopal,8500)

(6,Komal,22,MP,4500,6,Komal,22,MP,4500)

(7,Muffy,24,Indore,10000,7,Muffy,24,Indore,10000)

INNER JOIN:

coustomer_orders = JOIN customers BY id, orders BY customer_id;

Dump coustomer_orders;

INPUT:

1,Ramesh,32,Ahmedabad,2000.00

2,Khilan,25,Delhi,1500.00

3,kaushik,23,Kota,2000.00

4,Chaitali,25,Mumbai,6500.00

5,Hardik,27,Bhopal,8500.00

6,Komal,22,MP,4500.00

7,Muffy,24,Indore,10000.00

OUTPUT:

(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)
(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)

RIGHT OUTER JOIN:

outer_right = JOIN customers BY id RIGHT, orders BY customer_id;
Dump outer_right

INPUT:

1,Ramesh,32,Ahmedabad,2000.00
2,Khilan,25,Delhi,1500.00
3,kaushik,23,Kota,2000.00
4,Chaitali,25,Mumbai,6500.00
5,Hardik,27,Bhopal,8500.00
6,Komal,22,MP,4500.00
7,Muffy,24,Indore,10000.00

OUTPUT:

(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)
(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)

LEFT OUTER JOIN:

outer_left = JOIN customers BY id LEFT OUTER, orders BY customer_id;
Dump outer_left;

INPUT:

1,Ramesh,32,Ahmedabad,2000.00
2,Khilan,25,Delhi,1500.00
3,kaushik,23,Kota,2000.00
4,Chaitali,25,Mumbai,6500.00
5,Hardik,27,Bhopal,8500.00
6,Komal,22,MP,4500.00
7,Muffy,24,Indore,10000.00

OUTPUT:

(1,Ramesh,32,Ahmedabad,2000,,,,)

(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)

(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)

(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)

(5,Hardik,27,Bhopal,8500,,,,)

(6,Komal,22,MP,4500,,,,)

(7,Muffy,24,Indore,10000,,,,)

FULL OUTER JOIN:

outer_full = JOIN customers BY id FULL OUTER, orders BY customer_id;

Dump outer_full;

INPUT:

1,Ramesh,32,Ahmedabad,2000.00

2,Khilan,25,Delhi,1500.00

3,kaushik,23,Kota,2000.00

4,Chaitali,25,Mumbai,6500.00

5,Hardik,27,Bhopal,8500.00

6,Komal,22,MP,4500.00

7,Muffy,24,Indore,10000.00

OUTPUT:

(1,Ramesh,32,Ahmedabad,2000,,,,)

(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)

(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)

(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)

(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)

(5,Hardik,27,Bhopal,8500,,,,)

(6,Komal,22,MP,4500,,,,)

(7,Muffy,24,Indore,10000,,,,)

17. Write a pig script to implement the ORDER BY operator.

```
student_details = LOAD 'home/cse/Desktop/pig_data/student_details.txt' USING PigStorage(',') as
(id:int, firstname:chararray, lastname:chararray,age:int, phone:chararray, city:chararray);
order_by_data = ORDER student_details BY age DESC;
Dump order_by_data;
```

INPUT:

```
001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddharth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanty,23,9848022336,Bhubaneswar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivandrum
008,Bharathi,Nambiayar,24,9848022333,Chennai
```

OUTPUT:

```
(8,Bharathi,Nambiayar,24,9848022333,Chennai)
(7,Komal,Nayak,24,9848022334,trivandrum)
(6,Archana,Mishra,23,9848022335,Chennai)
(5,Trupthi,Mohanty,23,9848022336,Bhubaneswar)
(3,Rajesh,Khanna,22,9848022339,Delhi)
(2,siddharth,Battacharya,22,9848022338,Kolkata)
(4,Preethi,Agarwal,21,9848022330,Pune)
(1,Rajiv,Reddy,21,9848022337,Hyderabad)
```