EX 1 - **Perform setting up and Installing Hadoop in its three operating modes: Standalone, Pseudo distributed, fully distributed**

$ sudo apt installopenjdk-8-jdk -y

$ java -version

$ sudo adduser hadoop

$ sudo usermod -aG sudo hadoop

$ sudo su – Hadoop

$ apt install openssh-server openssh-client -y

$ sudo su - hadoop

$ ssh-keygen -t rsa

$ sudo cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

$ sudo chmod 640 ~/.ssh/authorized\_keys

$ ssh localhost

$ sudo su - hadoop

$wget <https://downloads.apache.org/hadoop/common/hadoop3.3.1/hadoop-3.3.1.tar.gz>

$ tar -xvzf hadoop-3.3.1.tar.gz

$ sudo mv hadoop-3.3.1 /usr/local/hadoop

$ sudo mkdir /usr/local/hadoop/logs

$ sudo chown -R hadoop:hadoop /usr/local/Hadoop

$ sudo nano ~/.bashrc

Add the following lines to the file. Save and close the file.

export HADOOP\_HOME=/usr/local/hadoop

export HADOOP\_INSTALL=$HADOOP\_HOME

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

export HADOOP\_COMMON\_HOME=$HADOOP\_HOME

export HADOOP\_HDFS\_HOME=$HADOOP\_HOME

export YARN\_HOME=$HADOOP\_HOME

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native

export PATH=$PATH:$HADOOP\_HOME/sbin:$HADOOP\_HOME/bin

export HADOOP\_OPTS="-Djava.library.path=$HADOOP\_HOME/lib/native"

$ source ~/.bashrc

$ readlink -f /usr/bin/javac

$ sudo nano $HADOOP\_HOME/etc/hadoop/hadoop-env.sh

JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64

export HADOOP\_CLASSPATH+=" $HADOOP\_HOME/lib/\*.jar"

* $ cd /usr/local/hadoop/lib
* $ sudo wget <https://jcenter.bintray.com/javax/activation/javax.activation-api/1.2.0/javax.activation-api-1.2.0.jar>
* $ hadoop version
* $ sudo nano $HADOOP\_HOME/etc/hadoop/core-site.xml

<configuration>

<property>

<name>fs.default.name</name>

<value>hdfs://0.0.0.0:9000</value>

<description>The default file system URI</description>

</property>

</configuration>

* $ sudo mkdir -p /home/hadoop/hdfs/{namenode,datanode}
* $ sudo nano $HADOOP\_HOME/etc/hadoop/hdfs-site.xml

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.name.dir</name>

<value>file:///home/hadoop/hdfs/namenode</value>

</property>

<property>

<name>dfs.data.dir</name>

<value>file:///home/hadoop/hdfs/datanode</value>

</property>

</configuration>

$ sudo nano $HADOOP\_HOME/etc/hadoop/mapred-site.xml

**<configuration>**

**<property>**

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

**<value>yarn</value>**

**</property>**

**</configuration>**

$ sudo nano $HADOOP\_HOME/etc/hadoop/yarn-site.xml

**<configuration>**

**<property>**

**<name>yarn.nodemanager.aux-services</name>**

**<value>mapreduce\_shuffle</value>**

**</property>**

**</configuration>**

$ sudo su - hadoop

$ hdfs namenode -format

$ start-dfs.sh

$ start-yarn.sh

$ jps

<http://localhost:9870/9864/8088/>

**EX-2 Implement the following file management tasks in Hadoop**

hdfs dfs -mkdir /demo/

hdfs dfs -ls /

hdfs dfs -put ‘/home/Hadoop/lab/dem’ /demo/

hadoop fs -get <hdfs\_src> <localdst>

hadoop fs-cat /demo/demo.txt

hadoop fs-cp /demo/demo.txt /demo2

hdfs fs -mv /demo/lab /newdir/

hdfs dfs -rmdir /lab/

hadoop fs -du /demo/lab

**EX-3 WORDCOUNT**

Create new folder WordCount

Upload the input.txt file in the WordCount folder

Upload the WordCount.java file in the WordCount folder

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.mapreduce.Job;

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

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

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

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

import org.apache.hadoop.fs.Path;

public class WordCount

{

public static class Map extends Mapper<LongWritable,Text,Text,IntWritable> {

public void map(LongWritable key, Text value,Context context) throws

IOException,InterruptedException{

String line = value.toString();

StringTokenizer tokenizer = new StringTokenizer(line);

while (tokenizer.hasMoreTokens()) {

value.set(tokenizer.nextToken());

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

}

}

}

public static class Reduce extends Reducer<Text,IntWritable,Text,IntWritable> {

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

throws IOException,InterruptedException {

int sum=0;

for(IntWritable x: values)

{

sum+=x.get();

}

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

}

}

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

Configuration conf= new Configuration();

Job job = new Job(conf,"My Word Count Program");

job.setJarByClass(WordCount.class);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

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

FileInputFormat.addInputPath(job, new Path(args[0]));

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

outputPath.getFileSystem(conf).delete(outputPath);

System.exit(job.waitForCompletion(true) ? 0 : 1);

}}

Create classes folder inside the WordCount folder

Click on web shell to open

Login into web shell

Move to the WordCount folder

Set the path for java file using the below commands,

* export HADOOP\_CLASSPATH=$(hadoop classpath)
* echo $HADOOP\_CLASSPATH

Compile the java file using the below command

* javac -classpath ${HADOOP\_CLASSPATH} -d '/home/kct5thsemcdhid02/WordCount/classes' '/home/kct5thsemcdhid02/WordCount/WordCount.java'

Create the jar file using the below command

* jar -cvf WordCount.jar -C '/home/kct5thsemcdhid02/WordCount/classes'/ .

Create a WordCount directory in Hadoop, inside the WordCount directory create input and output directory, the following are the commands used

* hdfs dfs -mkdir /user/kct5thsemcdhid02/WordCount
* hdfs dfs -mkdir /user/kct5thsemcdhid02/WordCount/input
* hdfs dfs -mkdir /user/kct5thsemcdhid02/WordCount/output

Put the input file from the local system to input directory in Hadoop

* hdfs dfs -put '/home/kct5thsemcdhid02/WordCount/input.txt' '/user/kct5thsemcdhid02/WordCount/input'

To check the input file is in Hadoop use cat command

* hdfs dfs -cat /user/kct5thsemcdhid02/WordCount/input/\*

Run the map reduce program by the following command

* hadoop jar /home/kct5semcdhid02/WordCount/WordCount.jar’ WordCount /user/kct5thsemcdhid02/WordCount/input /user/kct5thsemcdhid02/WordCount/output

Verify the output by using the following command

* hdfs dfs -cat /user/kct5thsemcdhid02/WordCount/output/\*

**EX-4 MATRIX**

import java.io.IOException;

import java.util.\*;

import java.util.AbstractMap.SimpleEntry;

import java.util.Map.Entry;

import org.apache.hadoop.fs.Path;

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

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

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

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

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

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

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

public class MatrixMultiplication {

public static class Map

extends org.apache.hadoop.mapreduce.Mapper<LongWritable, Text, Text, Text> {

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

            throws IOException, InterruptedException {

        Configuration conf = context.getConfiguration();

        int m = Integer.parseInt(conf.get("m"));

        int p = Integer.parseInt(conf.get("p"));

        String line = value.toString();

        String[] indicesAndValue = line.split(",");

        Text outputKey = new Text();

        Text outputValue = new Text();

        if (indicesAndValue[0].equals("M")) {

            for (int k = 0; k < p; k++) {

                outputKey.set(indicesAndValue[1] + "," + k);

                outputValue.set(indicesAndValue[0] + "," + indicesAndValue[2]

                        + "," + indicesAndValue[3]);

                context.write(outputKey, outputValue);

            }        } else {

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

                outputKey.set(i + "," + indicesAndValue[2]);

                outputValue.set("N," + indicesAndValue[1] + ","

                        + indicesAndValue[3]);

                context.write(outputKey, outputValue);

            }        }    }}

public static class Reduce

extends org.apache.hadoop.mapreduce.Reducer<Text, Text, Text, Text> {

    @Override

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

            throws IOException, InterruptedException {

        String[] value;

        HashMap<Integer, Float> hashA = new HashMap<Integer, Float>();

        HashMap<Integer, Float> hashB = new HashMap<Integer, Float>();

        for (Text val : values) {

            value = val.toString().split(",");

            if (value[0].equals("M")) {

                hashA.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));

            } else {

                hashB.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));

            }        }

        int n = Integer.parseInt(context.getConfiguration().get("n"));

        float result = 0.0f;

        float m\_ij;

        float n\_jk;

        for (int j = 0; j < n; j++) {

            m\_ij = hashA.containsKey(j) ? hashA.get(j) : 0.0f;

            n\_jk = hashB.containsKey(j) ? hashB.get(j) : 0.0f;

            result += m\_ij \* n\_jk;

        }

        if (result != 0.0f) {

            context.write(null,

                    new Text(key.toString() + "," + Float.toString(result)));

        }    }}

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

    if (args.length != 2) {

System.err.println("Usage: MatrixMultiplication <in\_dir> <out\_dir>");

System.exit(2); }

    Configuration conf = new Configuration();

// M is an m-by-n matrix; N is an n-by-p matrix.

conf.set("m", "1000");

conf.set("n", "100");

conf.set("p", "1000");

@SuppressWarnings("deprecation")

        Job job = new Job(conf, "MatrixMultiplication");

job.setJarByClass(MatrixMultiplication.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

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

job.waitForCompletion(true); }}

Create classes folder inside the matrix folder

move to the matrix folder in the web shell

* Cd matrix

Set the path for java file

* export HADOOP\_CLASSPATH=$(hadoop classpath)
* echo $HADOOP\_CLASSPATH

Compile the java file

* javac -classpath ${HADOOP\_CLASSPATH} -d '/home/kct5thsemcdhid02/matrix/classes' '/home/kct5thsemcdhid02/matrix/MatrixMultiplication.java'

Create the jar file

* jar -cvf MatrixMultiplication.jar -C '/home/kct5thsemcdhid02/matrix/classes'/ .

Create a matrix directory in Hadoop, inside the matrix directory create input and output directory, the following are the commands used

* hdfs dfs -mkdir /user/kct5thsemcdhid02/matrix
* hdfs dfs -mkdir /user/kct5thsemcdhid02/matrix/input
* hdfs dfs -mkdir /user/kct5thsemcdhid02/matrix/output

Put the input file from the local system to input directory in Hadoop

* hdfs dfs -put '/home/kct5thsemcdhid02/matrix/input.txt' '/user/kct5thsemcdhid02/matrix/input'

To check the input file is in Hadoop use cat command

* hdfs dfs -cat /user/kct5thsemcdhid02/matrix/input/\*

Run the map reduce program by the following command

* hadoop jar /home/kct5thsemcdhid02/matrix/MatrixMultiplication.jar MatrixMultiplication /user/kct5thsemcdhid02/matrix/input /user/kct5thsemcdhid02/matrix/output

Verify the output by using the following command

* hdfs dfs -cat /user/kct5thsemcdhid02/matrix/output/\*

**LAB 5 MIN/MAX TEMP**

Create new folder weather

Upload the input.txt and MinMaxTemperature.java file and create classes directory in the weather directory

import java.io.IOException;

import java.util.StringTokenizer;

import java.text.DecimalFormat;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

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

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

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.mapreduce.lib.output.TextOutputFormat;

public class MinMaxTemperature {

public static String calOutputName = "California";

public static String nyOutputName = "Newyork";

public static String njOutputName = "Newjersy";

public static String ausOutputName = "Austin";

public static String bosOutputName = "Boston";

public static String balOutputName = "Baltimore";

public static class WhetherForcastMapper extends Mapper<Object, Text, Text, Text> {

public void map(Object keyOffset, Text dayReport, Context con) throws IOException, InterruptedException {

StringTokenizer strTokens = new StringTokenizer(dayReport.toString(),"\t");

int counter = 0;

Float currnetTemp = null;

Float minTemp = Float.MAX\_VALUE;

Float maxTemp = Float.MIN\_VALUE;

String date = null;

String currentTime = null;

String minTempANDTime = null;

String maxTempANDTime = null;

while (strTokens.hasMoreElements()) {

if (counter == 0) {

date = strTokens.nextToken();

} else {

if (counter % 2 == 1) {

currentTime = strTokens.nextToken();

}

else {

currnetTemp = Float.parseFloat(strTokens.nextToken());

if (minTemp > currnetTemp) {

minTemp = currnetTemp;

minTempANDTime = minTemp + "AND" + currentTime;

}

if (maxTemp < currnetTemp) {

maxTemp = currnetTemp;

maxTempANDTime = maxTemp + "AND" + currentTime;

} } } counter++; }

Text temp = new Text();

temp.set(maxTempANDTime);

Text dateText = new Text();

dateText.set(date);

try {

con.write(dateText, temp);

} catch (Exception e) {

e.printStackTrace();

}

temp.set(minTempANDTime);

dateText.set(date);

con.write(dateText, temp); }}

public static class WhetherForcastReducer extends Reducer<Text, Text, Text, Text> {

MultipleOutputs<Text, Text> mos;

public void setup(Context context) {

mos = new MultipleOutputs<Text, Text>(context);

}

public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

int counter = 0;

String reducerInputStr[] = null;

String f1Time = "";

String f2Time = "";

String f1 = "", f2 = "";

Text result = new Text();

for (Text value : values) {

if (counter == 0) {

reducerInputStr = value.toString().split("AND");

f1 = reducerInputStr[0];

f1Time = reducerInputStr[1]; }

else {

reducerInputStr = value.toString().split("AND");

f2 = reducerInputStr[0];

f2Time = reducerInputStr[1]; }

counter = counter + 1; }

if (Float.parseFloat(f1) > Float.parseFloat(f2)) {

result = new Text("Time: " + f2Time + " MinTemp: " + f2 + "\t" + "Time: " + f1Time + " MaxTemp: " + f1);

} else {

result = new Text("Time: " + f1Time + " MinTemp: " + f1 + "\t" + "Time: " + f2Time + " MaxTemp: " + f2); }

String fileName = "";

if (key.toString().substring(0, 2).equals("CA")) {

fileName = MinMaxTemperature.calOutputName;

} else if (key.toString().substring(0, 2).equals("NY")) {

fileName = MinMaxTemperature.nyOutputName;

} else if (key.toString().substring(0, 2).equals("NJ")) {

fileName = MinMaxTemperature.njOutputName;

} else if (key.toString().substring(0, 3).equals("AUS")) {

fileName = MinMaxTemperature.ausOutputName;

} else if (key.toString().substring(0, 3).equals("BOS")) {

fileName = MinMaxTemperature.bosOutputName;

} else if (key.toString().substring(0, 3).equals("BAL")) {

fileName = MinMaxTemperature.balOutputName;}

String strArr[] = key.toString().split("\_");

key.set(strArr[1]);

mos.write(fileName, key, result); }

public void cleanup(Context context) throws IOException,InterruptedException { mos.close(); }}

public static void main(String[] args) throws IOException,

ClassNotFoundException, InterruptedException {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Wheather Statistics of USA");

job.setJarByClass(MinMaxTemperature.class);

job.setMapperClass(WhetherForcastMapper.class);

job.setReducerClass(WhetherForcastReducer.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

MultipleOutputs.addNamedOutput(job, calOutputName,TextOutputFormat.class, Text.class, Text.class);

MultipleOutputs.addNamedOutput(job, nyOutputName,TextOutputFormat.class, Text.class, Text.class);

MultipleOutputs.addNamedOutput(job, njOutputName,TextOutputFormat.class, Text.class, Text.class);

MultipleOutputs.addNamedOutput(job, bosOutputName,TextOutputFormat.class, Text.class, Text.class);

MultipleOutputs.addNamedOutput(job, ausOutputName,TextOutputFormat.class, Text.class, Text.class);

MultipleOutputs.addNamedOutput(job, balOutputName,TextOutputFormat.class, Text.class, Text.class);

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

FileInputFormat.addInputPath(job, new Path(args[0]));

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

outputPath.getFileSystem(conf).delete(outputPath);

try {

System.exit(job.waitForCompletion(true) ? 0 : 1);

} catch (Exception e) {

e.printStackTrace(); }}}

Move to weather directory

* cd weather

Set the path for java file using the below commands,

* export HADOOP\_CLASSPATH=$(hadoop classpath)
* echo $HADOOP\_CLASSPATH

Compile the java file using the below command

* javac -classpath ${HADOOP\_CLASSPATH} -d '/home/kct5thsemcdhid02/weather/classes' '/home/kct5thsemcdhid02/weather/MinMaxTemperature.java'

Create the jar file using the below command

* jar -cvf MinMaxTemperature.jar -C '/home/kct5thsemcdhid02/weather/classes'/ .

Create a weather directory in Hadoop, inside the weather directory create input and output directory, the following are the commands used

* hdfs dfs -mkdir /user/kct5thsemcdhid02/weather
* hdfs dfs -mkdir /user/kct5thsemcdhid02/weather/input
* hdfs dfs -mkdir /user/kct5thsemcdhid02/weather/output

Put the input file from the local system to input directory in Hadoop

* hdfs dfs -put '/home/kct5thsemcdhid02/weather/input.txt' '/user/kct5thsemcdhid02/weather/input'

To check the input file is in Hadoop use cat command

* hdfs dfs -cat /user/kct5thsemcdhid02/weather/input/\*

Run the map reduce program by the following command

* hadoop jar /home/kct5semcdhid02/weather/MinMaxTemperature.jar’ MinMaxTemperature /user/kct5thsemcdhid02/weather/input /user/kct5thsemcdhid02/weather/output

Verify the output by using the following command

* hdfs dfs -cat /user/kct5thsemcdhid02/weather/output/\*