
Group B

Assignment No: 2

Theory:

- **Steps to Install Hadoop for distributed environment**
- **Java Code for processes a log file of a system**

Steps to Install Hadoop for distributed environment:

Initially create one folder logfiles1 on desktop. In that folder store input file (access_log_short.csv), SalesMapper.java, SalesCountryReducer.java, SalesCountryDriver.java files)

Step 1) Go to Hadoop home directory and format the NameNode.

cd hadoop-2.7.3

bin/hadoop namenode -format

Step 2) Once the NameNode is formatted, go to hadoop-2.7.3/sbin directory and start all the daemons/nodes.

cd hadoop-2.7.3/sbin

1) Start NameNode:

The NameNode is the centerpiece of an HDFS file system. It keeps the directory tree of all files stored in the HDFS and tracks all the file stored across the cluster.

./hadoop-daemon.sh start namenode

2) Start DataNode:

On startup, a DataNode connects to the Namenode and it responds to the requests from the Namenode for different operations.

./hadoop-daemon.sh start datanode

3) Start ResourceManager:

ResourceManager is the master that arbitrates all the available cluster resources and thus helps in managing the distributed applications running on the YARN system. Its work is to manage each NodeManagers and the each application's ApplicationMaster.

./yarn-daemon.sh start resourcemanager

4) Start NodeManager:

The NodeManager in each machine framework is the agent which is responsible for managing containers, monitoring their resource usage and reporting the same to the ResourceManager.

./yarn-daemon.sh start nodemanager

5) Start JobHistoryServer:

JobHistoryServer is responsible for servicing all job history related requests from client.

./mr-jobhistory-daemon.sh start historyserver

Step 3) To check that all the Hadoop services are up and running, run the below command.

jps

Step 4) cd

Step 5) sudo mkdir mapreduce_vijay

Step 6) sudo chmod 777 -R mapreduce_vijay/

Step 7) sudo chown -R vijay mapreduce_vijay/

Step 8) sudo cp /home/vijay/Desktop/logfiles1/* ~/mapreduce_vijay/

Step 9) cd mapreduce_vijay/

Step 10) ls

Step 11) sudo chmod +r *.*

Step 12) export CLASSPATH="/home/vijay/hadoop-2.7.3/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.7.3.jar:/home/vijay/hadoop-2.7.3/share/hadoop/mapreduce/hadoop-mapreduce-client-common-2.7.3.jar:/home/vijay/hadoop-2.7.3/share/hadoop/common/hadoop-common-2.7.3.jar:~/mapreduce_vijay/SalesCountry/*:$HADOOP_HOME/lib/*"

Step 13) javac -d . SalesMapper.java SalesCountryReducer.java SalesCountryDriver.java

Step 14) ls

Step 15) cd SalesCountry/

Step 16) ls (check is class files are created)

Step 17) cd ..

Step 18) gedit Manifest.txt
(add following lines to it:
Main-Class: SalesCountry.SalesCountryDriver)

Step 19) jar -cfm mapreduce_vijay.jar Manifest.txt SalesCountry/*.class

Step 20) ls

Step 21) cd

Step 22) cd mapreduce_vijay/

Step 23) sudo mkdir /input200

Step 24) sudo cp access_log_short.csv /input200

Step 25) $HADOOP_HOME/bin/hdfs dfs -put /input200 /

Step 26) $HADOOP_HOME/bin/hadoop jar mapreduce_vijay.jar /input200 /output200

Step 27) hadoop fs -ls /output200

Step 28) hadoop fs -cat /out321/part-00000

Step 29) Now open the Mozilla browser and go to **localhost:50070/dfshealth.html** to check the NameNode interface.

Java Code to process logfile

Mapper Class:

```
package SalesCountry;

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.mapred.*;

public class SalesMapper extends MapReduceBase implements Mapper<LongWritable,
Text, Text, IntWritable> {
      private final static IntWritable one = new IntWritable(1);

      public void map(LongWritable key, Text value, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {

            String valueString = value.toString();
            String[] SingleCountryData = valueString.split("-");
            output.collect(new Text(SingleCountryData[0]), one);
      }
}
```

Reducer Class:

```
package SalesCountry;
```

```
import java.io.IOException;
import java.util.*;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class SalesCountryReducer extends MapReduceBase implements Reducer<Text,
IntWritable, Text, IntWritable> {

      public    void    reduce(Text    t_key,    Iterator<IntWritable>    values,
OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException
{
            Text key = t_key;
            int frequencyForCountry = 0;
            while (values.hasNext()) {
                  // replace type of value with the actual type of our value
                  IntWritable value = (IntWritable) values.next();
                  frequencyForCountry += value.get();

            }
            output.collect(key, new IntWritable(frequencyForCountry));
      }
}
```

Driver Class:

package SalesCountry;

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

public class SalesCountryDriver {
      public static void main(String[] args) {
            JobClient my_client = new JobClient();
            // Create a configuration object for the job
            JobConf job_conf = new JobConf(SalesCountryDriver.class);
```

```
            // Set a name of the Job
            job_conf.setJobName("SalePerCountry");

            // Specify data type of output key and value
            job_conf.setOutputKeyClass(Text.class);
            job_conf.setOutputValueClass(IntWritable.class);

            // Specify names of Mapper and Reducer Class
            job_conf.setMapperClass(SalesCountry.SalesMapper.class);
            job_conf.setReducerClass(SalesCountry.SalesCountryReducer.class);

            // Specify formats of the data type of Input and output
            job_conf.setInputFormat(TextInputFormat.class);
            job_conf.setOutputFormat(TextOutputFormat.class);

            // Set input and output directories using command line arguments,
            //arg[0] = name of input directory on HDFS, and arg[1] =  name of
output directory to be created to store the output file.

            FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
            FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));

            my_client.setConf(job_conf);
            try {
                  // Run the job
                  JobClient.runJob(job_conf);
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }
}
```

Input File

Pune

Mumbai

Nashik

Pune

Nashik

Kolapur

Assignment Questions

1. Write down the steps for Design a distributed application using MapReduce which processes a log file of a system.