EXPERIMENT 11

TITLE: Design a distributed application using MapReduce

OBJECTIVE:

1. To explore different Big data processing techniques with use cases.
2. To study detailed concept of Map-Reduced.

SOFTWARE REQUIREMENTS:

1. Ubuntu 14.04 / 14.10
2. GNU C Compiler
3. Hadoop
4. Java

PROBLEM STATEMENT:- Design a distributed application using MapReduce which processes a log file of a system. List out the users who have logged for maximum period on the system. Use simple log file from the Internet and process it using a pseudo distribution mode on Hadoop platform.

THEORY:

Introduction

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner. MapReduce is a processing technique and a program model for distributed computing based on java.

The MapReduce algorithm contains two important tasks, namely Map and Reduce.Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).

Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes.

Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
Mapstage : The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

Reduce stage : This stage is the combination of the Shuffle stage and the Reduce stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

Inserting Data into HDFS:

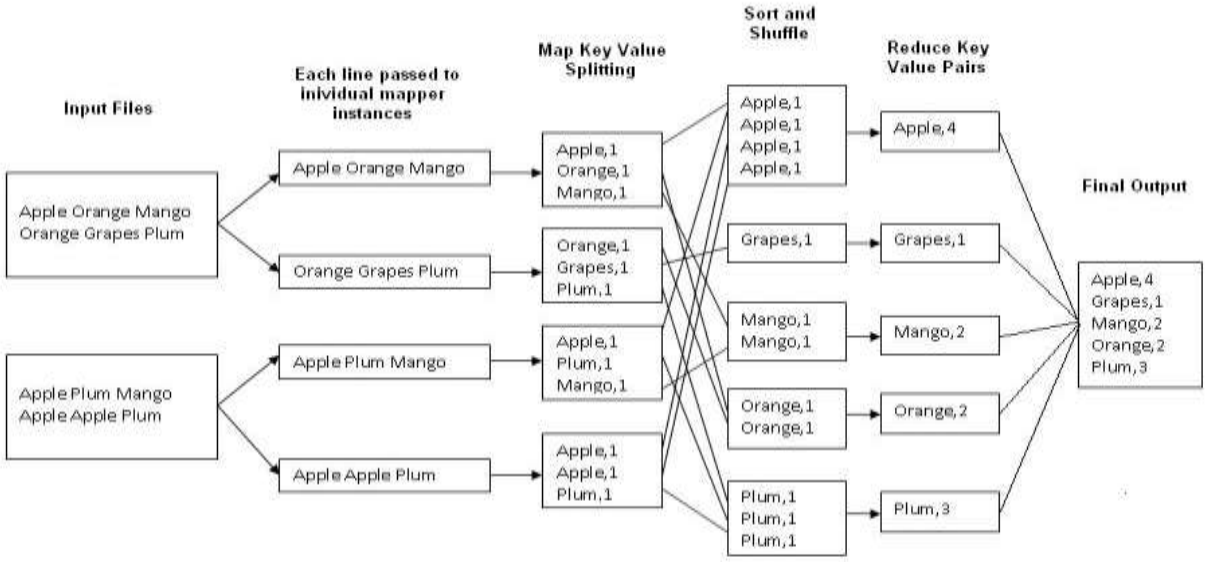

Fig.1 : An Example Program to Understand working of MapReduce Program.

•The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

• The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework.

•Input and Output types of a MapReduce job: (Input) <k1,v1> -> map -> <k2, v2>-> reduce -> <k3, v3> (Output).

Steps for Compilation & Execution of Program:

#sudo mkdir analyzelogs

ls

#sudo chmod -R 777 analyzelogs/

cd

ls

cd ..

pwd

ls

cd

pwd

#sudo chown -R hduser analyzelogs/

cd

ls

#cd analyzelogs/

ls

cd ..

Copy the Files (Mapper.java,Reduce.java,Driver.java to Analyzelogs Folder)

#sudo cp /home/mde/Desktop/count_logged_users/* -/analyzelogs/

Start HADOOP

#start-dfs.sh

#start-yarn.sh

#jps

cd

cd analyzelogs

ls

pwd

ls

#ls -ltr

```
#ls -al
#sudo chmod +r *.*
        pwd
#export    CLASSPATH="$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.9.0.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-common-2.9.0.jar:$HADOOP_HOME/share/hadoop/common/hadoop-common-2.9.0.jar:~/analyzelogs/SalesCountry/*:$HADOOP_HOME/lib/*"
```

Compile Java Files

```
# javac -d . SalesMapper.java SalesCountryReducer.java SalesCountryDriver.java
   ls
 #cd SalesCountry/
   ls
  cd ..
  #sudo gedit Manifest.txt
  #jar -cfm analyzelogs.jar Manifest.txt SalesCountry/*.class
    ls
   cd
   jps
#cd analyzelogs/
```

Create Directory on Hadoop

```
#sudo mkdir ~/input2000
    ls
   pwd
 #sudo cp access_log_short.csv ~/input2000/
# $HADOOP_HOME/bin/hdfs dfs -put ~/input2000 /
 # $HADOOP_HOME/bin/hadoop jar analyzelogs.jar /input2000 /output2000
```

```
# $HADOOP_HOME/bin/hdfs dfs -cat /output2000/part-00000

# stop-all.sh
 # jps
```

CONCLUSION: Thus we have learnt how to design a distributed application using MapReduce and process a log file of a system.