|  |  |
| --- | --- |
| **EX.NO: 10a** | **IMPLENTATION OF MAP AND REDUCE TASKS** |
| **DATE:** |

**INTRODUCTION**

In Hadoop, MapReduce is a computation that decomposes large manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers. The results of tasks can be joined together to compute final results.

MapReduce consists of 2 steps:

• Map Function – It takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (Key-Value pair).

Example – (Map function in Word Count)

Input Set of data

Bus, Car, bus, car, train, car, bus, car, train, bus, TRAIN,BUS, buS, caR, CAR, car, BUS, TRAIN

Output Convert into another set of data

(Key,Value) (Bus,1), (Car,1), (bus,1), (car,1), (train,1),

(car,1), (bus,1), (car,1), (train,1), (bus,1),

(TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1),

(car,1), (BUS,1), (TRAIN,1)

• Reduce Function – Takes the output from Map as an input and combines those data tuples into a smaller set of tuples.

Example – (Reduce function in Word Count)

Input

(output of Map function) Set of Tuples

(Bus,1), (Car,1), (bus,1), (car,1), (train,1),

(car,1), (bus,1), (car,1), (train,1), (bus,1),

(TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1),

(car,1), (BUS,1), (TRAIN,1)

Output Converts into smaller set of tuples (BUS,7),

(CAR,7),

(TRAIN,4)

**AIM:**

To Write a Word count program that demonstrate the use of Map and Reduce tasks.

**PROCEDURE:**

We will use the example mapreduce program to test the Hadoop.

**Step 1:** The first step is to create a directory called input in the home screen.

$ mkdir ~/input

**Step 2:** Copy the Hadoop configuration to that file.

$ cp /usr/local/hadoop/etc/hadoop/\*.xml ~/input

**Step 3:** Run the Mapreduce hadoop-mapreduce-examples. It is a java archieve with a lot of programs inside.

Use the grep program from the Mapdreduce program.

We are going to find the occurrence of 'principal' at the end or before the declarative.

Since the expression is case sensitive, we could not find if it is capitalized.

$/usr/local/hadoop/bin/hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.7.jar grep ~/input ~/grep\_example 'principal[.]\*'

**Output:**

File System Counters

FILE: Number of bytes read=1247674

FILE: Number of bytes written=2324248

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

Map-Reduce Framework

Map input records=2

Map output records=2

Map output bytes=37

Map output materialized bytes=47

Input split bytes=114

Combine input records=0

Combine output records=0

Reduce input groups=2

Reduce shuffle bytes=47

Reduce input records=2

Reduce output records=2

Spilled Records=4

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

GC time elapsed (ms)=61

Total committed heap usage (bytes)=263520256

Shuffle Errors

BAD\_ID=0

CONNECTION=0

IO\_ERROR=0

WRONG\_LENGTH=0

WRONG\_MAP=0

WRONG\_REDUCE=0

File Input Format Counters

Bytes Read=151

File Output Format Counters

Bytes Written=37

Step 4:

The results will be stored in the output directory and you can check it using cat.

$ cat ~/grep\_example/\*

**Output**

6 principal

1 principal.

The output indicates that the word principal was found after six non-occurrence.

**RESULT:**

Thus, the implementation of a Word count program that demonstrate the use of Map and Reduce tasks has been verified and executed successfully.

|  |  |
| --- | --- |
| **EX.NO: 10b** | **IMPLENTATION OF MAP AND REDUCE TASKS** |
| **DATE:** |

**AIM:**

To Write a Word count program that demonstrate the use of Map and Reduce tasks.

**PROCEDURE:**

Make sure that Hadoop is installed on your system with the Java SDK.

**Step 1:** Open Eclipse> File > New > Java Project >( Name it – MRProgramsDemo) > Finish.

**Step 2**: Right Click > New > Package ( Name it - PackageDemo) > Finish.

**Step 3:** Right Click on Package > New > Class (Name it - WordCount).

**Step 4:** Add Following Reference Libraries:

* 1. Right Click on Project > Build Path> Add External
     1. */usr/lib/hadoop-0.20/****hadoop-core.jar***
     2. *Usr/lib/hadoop-0.20/lib/****Commons-cli-1.2.jar***

**Step 5:** Type the following code:

package PackageDemo;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

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.Reducer;

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 c=new Configuration();

String[] files=new GenericOptionsParser(c,args).getRemainingArgs();

Path input=new Path(files[0]);

Path output=new Path(files[1]);

Job j=new Job(c,"wordcount");

j.setJarByClass(WordCount.class);

j.setMapperClass(MapForWordCount.class);

j.setReducerClass(ReduceForWordCount.class);

j.setOutputKeyClass(Text.class);

j.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(j, input);

FileOutputFormat.setOutputPath(j, output);

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

}

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

public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException

{

String line = value.toString();

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

for(String word: words )

{

Text outputKey = new Text(word.toUpperCase().trim());

IntWritable outputValue = new IntWritable(1);

con.write(outputKey, outputValue);

}

}

}

public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text, IntWritable>

{

public void reduce(Text word, Iterable<IntWritable> values, Context con) throws IOException, InterruptedException

{

int sum = 0;

for(IntWritable value : values)

{

sum += value.get();

}

con.write(word, new IntWritable(sum));

}

}

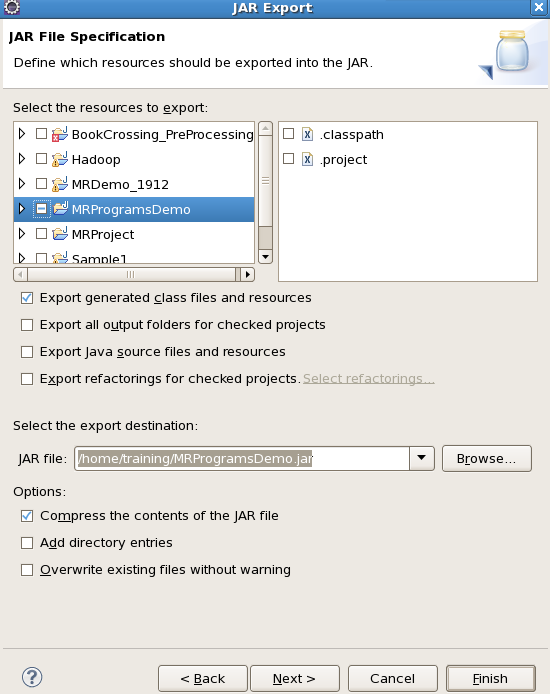
}

The above program consists of three classes:

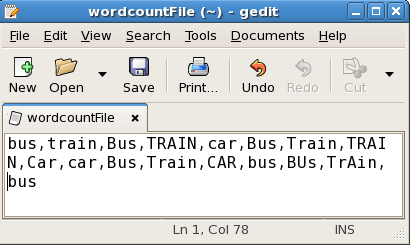
* Driver class (Public, void, static, or main; this is the entry point).
* The Map class which **extends** the public class Mapper<KEYIN,VALUEIN,KEYOUT,VALUEOUT>  and implements the Map function.
* The Reduce class which extends the public class Reducer<KEYIN,VALUEIN,KEYOUT,VALUEOUT> and implements the Reduce function.

**Step 6:** Make  a jar file

Right Click on Project> Export> Select export destination as **Jar File** > next> Finish.



**Step 7:** Take a text file and move it into HDFS format:



To move this into Hadoop directly, open the terminal and enter the following commands:

[training@localhost ~]$ hadoop fs -put wordcountFile wordCountFile

**Step 8**: Run the jar file:

*(Hadoop jar jarfilename.jar packageName.ClassName  PathToInputTextFile PathToOutputDirectry)*

[training@localhost ~]$ hadoop jar MRProgramsDemo.jar PackageDemo.WordCount wordCountFile MRDir1

**Step 9:** Open the result:

[training@localhost ~]$ hadoop fs -ls MRDir1

Found 3 items

-rw-r--r-- 1 training supergroup 0 2016-02-23 03:36 /user/training/MRDir1/\_SUCCESS

drwxr-xr-x - training supergroup 0 2016-02-23 03:36 /user/training/MRDir1/\_logs

-rw-r--r-- 1 training supergroup 20 2016-02-23 03:36 /user/training/MRDir1/part-r-00000

[training@localhost ~]$ hadoop fs -cat MRDir1/part-r-00000

BUS 7

CAR 4

TRAIN 6

**RESULT**:

Thus, the implementation of a Word count program that demonstrate the use of Map and Reduce tasks has been verified and executed successfully.

|  |  |
| --- | --- |
| **EX.NO: 11** | **MIGRATE A SINGLE INSTANCE TO ANOTHER COMPUTE HOST** |
| **DATE:** |

**AIM**

To find procedure to Migrate a single instance to another compute host

**PROCEDURE**

1. To list the VMs you want to migrate, run:

$nova list

1. After selecting a VM from the list, run this command where VM\_ID is set to the ID in the list returned in the previous step:

$nova show VM\_ID

1. Use the **nova migrate** command.

$nova migrate VM\_ID

1. To migrate an instance and watch the status, use this example script:

#!/bin/bash

# Provide usage

usage(){

echo"Usage: $0 VM\_ID"

exit 1

}

[[$# -eq0]]&& usage

# Migrate the VM to an alternate hypervisor

echo -n "Migrating instance to alternate host"

VM\_ID=$1

nova migrate $VM\_ID

VM\_OUTPUT=`nova show $VM\_ID`

VM\_STATUS=`echo"$VM\_OUTPUT"|grep status |awk'{print $4}'`

while[["$VM\_STATUS" !="VERIFY\_RESIZE"]];do

echo -n "."

sleep 2

VM\_OUTPUT=`nova show $VM\_ID`

VM\_STATUS=`echo"$VM\_OUTPUT"|grep status |awk'{print $4}'`

done

nova resize-confirm $VM\_ID

echo" instance migrated and resized."

echo;

# Show the details for the VM

echo"Updated instance details:"

nova show $VM\_ID

# Pause to allow users to examine VM details

read -p "Pausing, press <enter> to exit."

**RESULT**

Thus, the procedure to migrate a single instance to another compute host has been studied and verified successfully

|  |  |
| --- | --- |
| **EX.NO: 12** | **MIGRATE A SINGLE INSTANCE TO ANOTHER COMPUTE HOST** |
| **DATE:** |

**INTRODUCTION:**

FUSE (Filesystem in Userspace) enables you to write a normal user application as a bridge for a traditional filesystem interface.

The hadoop-hdfs-fuse package enables you to use your HDFS cluster as if it were a traditional filesystem on Linux. It is assumed that you have a working HDFS cluster and know the hostname and port that your NameNode exposes.

**AIM:**

To mount the one node Hadoop cluster using FUSE.

**PROCEDURE:**

**. To install fuse-dfs on Ubuntu systems:**

hdpuser@jiju-PC:~$ wget http://archive.cloudera.com/cdh5/one-click-install/trusty/amd64/cdh5-repository\_1.0\_all.deb

–2016-07-24 09:10:33– http://archive.cloudera.com/cdh5/one-click-install/trusty/amd64/cdh5-repository\_1.0\_all.deb

Resolving archive.cloudera.com (archive.cloudera.com)… 151.101.8.167

Connecting to archive.cloudera.com (archive.cloudera.com)|151.101.8.167|:80… connected.

HTTP request sent, awaiting response… 200 OK

Length: 3508 (3.4K) [application/x-debian-package]

Saving to: ‘cdh5-repository\_1.0\_all.deb’

100%[======================================>] 3,508 –.-K/s in 0.09s

2016-07-24 09:10:34 (37.4 KB/s) – ‘cdh5-repository\_1.0\_all.deb’ saved [3508/3508]

hdpuser@jiju-PC:~$ sudo dpkg -i cdh5-repository\_1.0\_all.deb

Selecting previously unselected package cdh5-repository.

(Reading database … 170607 files and directories currently installed.)

Preparing to unpack cdh5-repository\_1.0\_all.deb …

Unpacking cdh5-repository (1.0) …

Setting up cdh5-repository (1.0) …

gpg: keyring `/etc/apt/secring.gpg’ created

gpg: keyring `/etc/apt/trusted.gpg.d/cloudera-cdh5.gpg’ created

gpg: key 02A818DD: public key “Cloudera Apt Repository” imported

gpg: Total number processed: 1

gpg: imported: 1

2. hdpuser@jiju-PC:~$ sudo apt-get update

3. hdpuser@jiju-PC:~$ sudo apt-get install hadoop-hdfs-fuse

4. hdpuser@jiju-PC:~$ sudo mkdir -p /home/hdpuser/hdfs

[sudo] password for hdpuser:

5. hdpuser@jiju-PC:~$ sudo hadoop-fuse-dfs dfs://localhost:54310 /home/hdpuser/hdfs/

INFO /data/jenkins/workspace/generic-package-ubuntu64-14-04/CDH5.8.0-Packaging-Hadoop-2016-07-12\_15-43-10/hadoop-2.6.0+cdh5.8.0+1601-1.cdh5.8.0.p0.93~trusty/hadoop-hdfs-project/hadoop-hdfs/src/main/native/fuse-dfs/fuse\_options.c:164 Adding FUSE arg /home/hdpuser/hdfs/

6. hdpuser@jiju-PC:~$ ls /home/hdpuser/hdfs/

7. hdpuser@jiju-PC:~$ mkdir /home/hdpuser/hdfs/new

8. hdpuser@jiju-PC:~$ ls /home/hdpuser/hdfs/

new

9. hdpuser@jiju-PC:~$ mkdir /home/hdpuser/hdfs/example

10. hdpuser@jiju-PC:~$ ls -l /home/hdpuser/hdfs/

total 8

drwxr-xr-x 2 hdpuser 99 4096 Jul 24 15:28 example

drwxr-xr-x 2 hdpuser 99 4096 Jul 24 15:19 new

To Unmont the file system

Using umount command the filesystem can be unmounted.

hdpuser@jiju-PC:~$ sudo umount /home/hdpuser/hdfs

NOTE: You can now add a permanent HDFS mount which persists through reboots.

To add a system mount:

• Open /etc/fstab and add lines to the bottom similar to these: (sudo vi /etc/fstab)

hadoop-fuse-dfs#dfs://<name\_node\_hostname>:<namenode\_port> <mount\_point> fuse allow\_other,usetrash,rw 2 0

For example:

sudo hadoop-fuse-dfs#dfs://localhost:54310 /home/hdpuser/hdfs fuse allow\_other,usetrash,rw 2 0

• Test to make sure everything is working properly:

$ mount <mount\_point>hdpuser@jiju-PC:~$ sudo mount /home/hdpuser/hdfs

**RESULT:**

Thus, the procedure to mount the one node Hadoop cluster using FUSE has been done successfully.