**Practical – 1**

**Experiment 1 - To Implement WordCount problem using Hadoop MapReduce in Eclipse: (Without Combiner & With Combiner)**

**What is MapReduce ?**

A MapReduce is a data processing tool which is used to process the data parallelly in a distributed form. It was developed in 2004, on the basis of paper titled as "MapReduce: Simplified Data Processing on Large Clusters," published by Google.

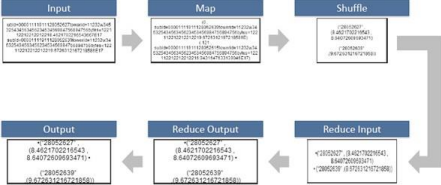
The MapReduce is a paradigm which has two phases, the mapper phase, and the reducer phase. In the Mapper, the input is given in the form of a key-value pair. The output of the Mapper is fed to the reducer as input. The reducer runs only after the Mapper is over. The reducer too takes input in key-value format, and the output of reducer is the final output.

**Steps in Map Reduce**

• The map takes data in the form of pairs and returns a list of <key, value> pairs. The keys will not be unique in this case.

• Using the output of Map, sort and shuffle are applied by the Hadoop architecture. This sort and shuffle acts on these list of <key, value> pairs and sends out unique keys and a list of values associated with this unique key <key, list(values)>.

• An output of sort and shuffle sent to the reducer phase. The reducer performs a defined function on a list of values for unique keys, and Final output <key, value> will be stored/displayed.



**Sort and Shuffle**

The sort and shuffle occur on the output of Mapper and before the reducer. When the Mapper task is complete, the results are sorted by key, partitioned if there are multiple reducers, and then written to disk. Using the input from each Mapper <k2,v2>, we collect all the values for each unique key k2. This output from the shuffle phase in the form of <k2, list(v2)> is sent as input to reducer phase.

**Data Flow In MapReduce**

MapReduce is used to compute the huge amount of data . To handle the upcoming data in a parallel and distributed form, the data has to flow from various phases.

**Phases of MapReduce data flow**

**Input reader**

The input reader reads the upcoming data and splits it into the data blocks of the appropriate size (64 MB to 128 MB). Each data block is associated with a Map function.

Once input reads the data, it generates the corresponding key-value pairs. The input files reside in HDFS.

**Map function**

The map function process the upcoming key-value pairs and generated the corresponding output key-value pairs. The map input and output type may be different from each other.

**Partition function**

The partition function assigns the output of each Map function to the appropriate reducer. The available key and value provide this function. It returns the index of reducers.

**Shuffling and Sorting**

The data are shuffled between/within nodes so that it moves out from the map and get ready to process for reduce function. Sometimes, the shuffling of data can take much computation time.

The sorting operation is performed on input data for Reduce function. Here, the data is compared using comparison function and arranged in a sorted form.

**Reduce function**

The Reduce function is assigned to each unique key. These keys are already arranged in sorted order. The values associated with the keys can iterate the Reduce and generates the corresponding output.

**Output writer**

Once the data flow from all the above phases, Output writer executes. The role of Output writer is to write the Reduce output to the stable storage.

**To Implement WordCount problem using Hadoop MapReduce in Eclipse:** Hadoop WordCount operation occurs in 3 stages –

➢ Mapper Phase

➢ Shuffle Phase

➢ Reducer Phase

**Hadoop WordCount - Mapper Phase Execution**

• The text from the input text file is tokenized into words to form a key value pair with all the words present in the input text file. The key is the word from the input file and value is ‘1’.

• For instance if you consider the sentence “An elephant is an animal”. • The mapper phase in the WordCount example will split the string into individual tokens i.e. words. In this case, the entire sentence will be split into 5 tokens (one for each word) with a value 1 as shown below –

Key-Value pairs from Hadoop Map Phase Execution-

(an,1)

(elephant,1)

(is,1)

(an,1)

(animal,1)

**Hadoop WordCount Example- Shuffle Phase Execution**

• After the map phase execution is completed successfully, shuffle phase is executed automatically wherein the key-value pairs generated in the map phase are taken as input and then sorted in alphabetical order.

• After the shuffle phase is executed from the WordCount example code, the output will look like this -

(an,1)

(an,1)

(animal,1)

(elephant,1)

(is,1)

**Hadoop WordCount Example- Reducer Phase Execution**

• In the reduce phase, all the keys are grouped together and the values for similar keys are added up to find the occurrences for a particular word.

• It is like an aggregation phase for the keys generated by the map phase. The reducer phase takes the output of shuffle phase as input and then reduces the key-value pairs to unique keys with values added up.

• In our example “An elephant is an animal.” is the only word that appears twice in the sentence.

• After the execution of the reduce phase of MapReduce WordCount example program, appears as a key only once but with a count of 2 as shown below -

(an,2)

(animal,1)

(elephant,1)

(is,1)

• This is how the MapReduce word count program executes and outputs the number of occurrences of a word in any given input file.

• An important point to note during the execution of the WordCount example is that the mapper class in the WordCount program will execute completely on the entire input file and not just a single sentence.

• Suppose if the input file has 15 lines then the mapper class will split the words of all the 15 lines and form initial key value pairs for the entire dataset.

• The reducer execution will begin only after the mapper phase is executed successfully.

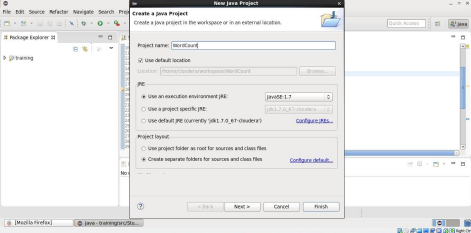
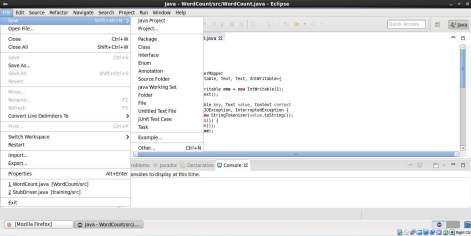
**Steps for Word Count in Cloudera: (Without Combiner)**

1) Open virtual box and then start cloudera quickstart

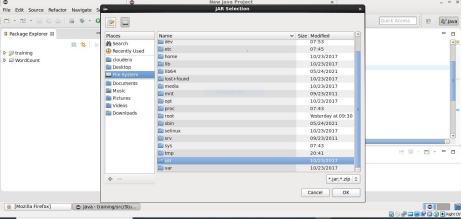
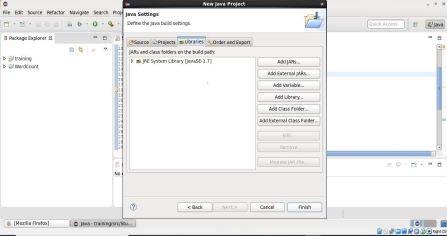
2) Open Eclipse present on the cloudera desktop

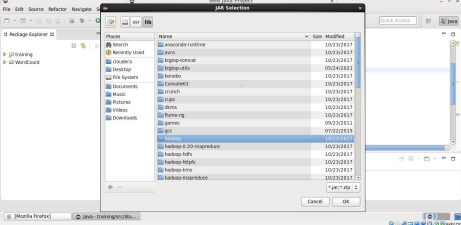
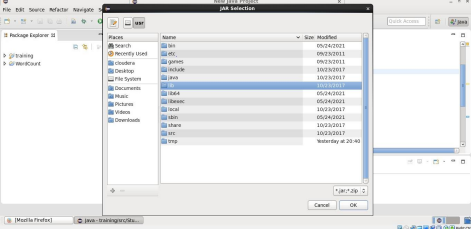


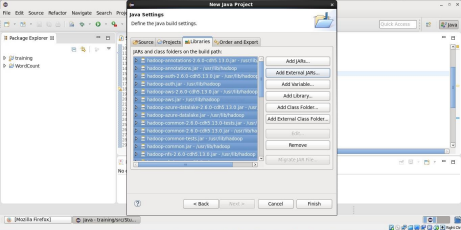
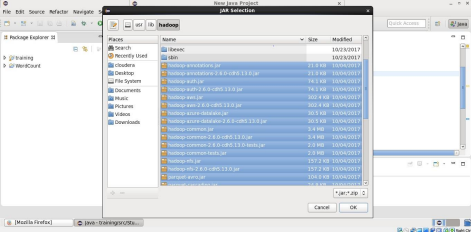
3) Create a new Java project clicking: File -> New -> Project -> Java Project -> Next (“WordCount” is the project name).

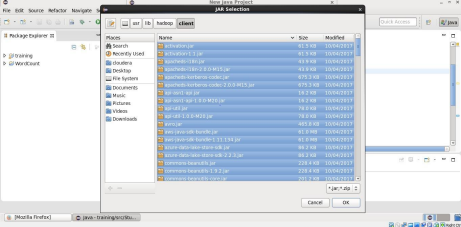
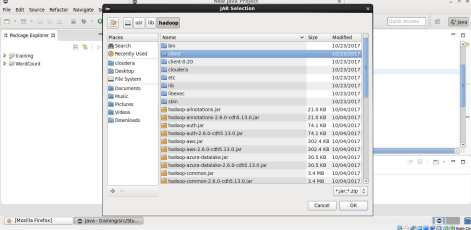


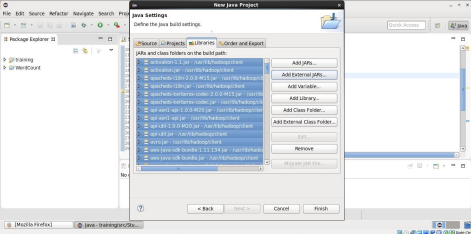
4) Adding the Hadoop libraries to the project Click on Libraries -> Add External JARs Click on File System -> usr -> lib -> hadoop Select all the libraries (JAR Files) -> click OK Click on Add External jars, -> client -> select all jar files -> ok -> Finish











5) Right Click on the name of Project “WordCount” -> New -> class Don’t write anything for package Write Name Textbox write “WordCount” -> Finish Then WordCount.java window will pop up







**Source code:**

**Packages**

import java.io.IOException;

import java.util.StringTokenizer;

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.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

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

**Mapper Logic**

public class WordCount {

public static 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 logic**

public static 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);

}

}

**Main function**

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

Configuration conf = new Configuration();

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);

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

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

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

}

}

**We are running the code without combiner. That is why we commented the combiner line in main function.**

6) Right Click on the project name WordCount -> Export -> Java -> JAR File -> Next -> for select the export destination for JAR file: browse -> Name : WordCount.jar -> save in folder -> cloudera -> Finish -> OK







7) Verify jar file from terminal by using Open terminal & type “ ls ” There it will show WordCount.jar **Check current working directory**

->pwd



8) We need to create an input file in local file system

Creating an input file named as ”abc.txt”.

Here listing all the directory present in hdfs using hdfs

dfs -ls / command

[cloudera@quickstart ~]$ hdfs dfs -ls /

Found 5 items

drwxr-xr-x - hbase supergroup 0 2023-01-18

21:12 /hbase

drwxr-xr-x - solr solr 0 2015-06-09 03:38

/solr

drwxrwxrwx - hdfs supergroup 0 2023-01-18

21:14 /tmp

drwxr-xr-x - hdfs supergroup 0 2015-06-09

03:38 /user

drwxr-xr-x - hdfs supergroup 0 2015-06-09

03:36 /var

9) Now we have to move this input file to hdfs. For this we create a direcory on hdfs using command hdfs dfs -mkdir /inputnew.

[cloudera@quickstart ~]$ hdfs dfs -mkdir /inputdir

[cloudera@quickstart ~]$ hdfs dfs -ls /

Found 6 items

drwxr-xr-x - hbase supergroup 0 2023-01-18

21:12 /hbase

drwxr-xr-x - cloudera supergroup 0 2023-01-18

22:03 /inputdir

drwxr-xr-x - solr solr 0 2015-06-09 03:38

/solr

drwxrwxrwx - hdfs supergroup 0 2023-01-18

21:14 /tmp

drwxr-xr-x - hdfs supergroup 0 2015-06-09

03:38 /user

drwxr-xr-x - hdfs supergroup 0 2015-06-09

~~03:36 /var~~

[cloudera@quickstart ~]$ hdfs dfs -put

-put: Not enough arguments: expected 1 but got 0 Usage: hadoop fs [generic options] -put [-f] [-p] [-l] <localsrc> ... <dst>

[cloudera@quickstart ~]$

[cloudera@quickstart ~]$

[cloudera@quickstart ~]$

[cloudera@quickstart ~]$

[cloudera@quickstart ~]$ hdfs dfs -put

/home/cloudera/Desktop/abc.txt/inputdir/

put: `/home/cloudera/Desktop/abc.txt/inputdir/': No such file or directory

[cloudera@quickstart ~]$ hdfs dfs -put

/home/cloudera/Desktop/abc.txt /inputdir/ put: `/home/cloudera/Desktop/abc.txt': No such file or directory

[cloudera@quickstart ~]$ hdfs dfs -put

/home/cloudera/Desktop/abc.txt

/inputdir/

put: `/home/cloudera/Desktop/abc.txt': No such file or directory

[cloudera@quickstart ~]$ ls

abc.txt Desktop eclipse kerberos

Pictures Videos

cloudera-manager Documents enterprise deployment.json lib

Public WordCount.jar

cm\_api.py Downloads express-deployment.json Music

Templates workspace

[cloudera@quickstart ~]$ ls Desktop

Eclipse.desktop Enterprise.desktop Express.desktop [cloudera@quickstart ~]$ cd Desktop

[cloudera@quickstart Desktop]$ gedit abc.txt [cloudera@quickstart Desktop]$ ls Desktop ls: cannot access Desktop: No such file or directory [cloudera@quickstart Desktop]$ ls

abc.txt Eclipse.desktop Enterprise.desktop Express.desktop

[cloudera@quickstart Desktop]$ hdfs dfs -put /home/cloudera/Desktop/abc.txt /inputdir/ [cloudera@quickstart Desktop]$ hdfs dfs -ls /inputdir Found 1 items

-rw-r--r-- 1 cloudera supergroup 289 2023-01-18 22:21 /inputdir/abc.txt

[cloudera@quickstart Desktop]$ hdfs dfs -ls /inputdir Found 1 items

-rw-r--r-- 1 cloudera supergroup 289 2023-01-18 22:21 /inputdir/abc.txt

[cloudera@quickstart Desktop]$ hdfs dfs -cat

/inputdir/abc.txt

Analyze Your Data

Hue is the open source web interface for Hadoop that lets you analyze

your data. Simply load in your data and then easily begin to analyze,

search, and visualize it. In the QuickStart VM, the administrative

username for Hue is 'cloudera' and the password is 'cloudera'.

[cloudera@quickstart Desktop]$ hadoop jar /home/cloudera/WordCount.jar

WordCount /inputdir/abc.txt /outputdir

23/01/18 22:36:40 INFO client.RMProxy: Connecting to ResourceManager

at /0.0.0.0:8032

23/01/18 22:36:41 WARN mapreduce.JobSubmitter: Hadoop command-line

option parsing not performed. Implement the Tool interface and execute

your application with ToolRunner to remedy this. 23/01/18 22:36:41 INFO input.FileInputFormat: Total input paths to process : 1

23/01/18 22:36:41 INFO mapreduce.JobSubmitter: number of splits:1

23/01/18 22:36:42 INFO mapreduce.JobSubmitter: Submitting tokens for

job: job\_1674105093530\_0002

23/01/18 22:36:42 INFO impl.YarnClientImpl: Submitted application

application\_1674105093530\_0002

23/01/18 22:36:42 INFO mapreduce.Job: The url to track the job:

http://quickstart.cloudera:8088/proxy/application\_1674 105093530\_0002/

23/01/18 22:36:42 INFO mapreduce.Job: Running job: job\_1674105093530\_0002

23/01/18 22:36:55 INFO mapreduce.Job: Job job\_1674105093530\_0002

running in uber mode : false

23/01/18 22:36:55 INFO mapreduce.Job: map 0% reduce 0%

23/01/18 22:37:05 INFO mapreduce.Job: map 100% reduce 0%

23/01/18 22:37:15 INFO mapreduce.Job: map 100% reduce 100%

23/01/18 22:37:15 INFO mapreduce.Job: Job job\_1674105093530\_0002

completed successfully

23/01/18 22:37:15 INFO mapreduce.Job: Counters: 49 File System Counters

FILE: Number of bytes read=588 FILE: Number of bytes written=221459 FILE: Number of read operations=0 FILE: Number of large read operations=0 FILE: Number of write operations=0 HDFS: Number of bytes read=402 HDFS: Number of bytes written=327 HDFS: Number of read operations=6 HDFS: Number of large read operations=0 HDFS: Number of write operations=2 Job Counters

Launched map tasks=1

Launched reduce tasks=1

Data-local map tasks=1

Total time spent by all maps in occupied slots (ms)=8456

Total time spent by all reduces in occupied slots (ms)=7017

Total time spent by all map tasks (ms)=8456 Total time spent by all reduce tasks (ms)=7017

Total vcore-seconds taken by all map tasks=8456

Total vcore-seconds taken by all reduce tasks=7017

Total megabyte-seconds taken by all map tasks=8658944

Total megabyte-seconds taken by all reduce tasks=7185408

Map-Reduce Framework

Map input records=3

Map output records=49

Map output bytes=484

Map output materialized bytes=588 Input split bytes=113

Combine input records=0

Combine output records=0

Reduce input groups=39

Reduce shuffle bytes=588

Reduce input records=49

Reduce output records=39

Spilled Records=98

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

GC time elapsed (ms)=213

CPU time spent (ms)=1330

Physical memory (bytes)

snapshot=342401024

Virtual memory (bytes)

snapshot=3007152128

Total committed heap usage

(bytes)=226365440

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=289

File Output Format Counters

Bytes Written=327

[cloudera@quickstart Desktop]$ hdfs dfs -ls /outputdir Found 2 items

-rw-r--r-- 1 cloudera supergroup 0 2023-01-18 22:37

/outputdir/\_SUCCESS

-rw-r--r-- 1 cloudera supergroup 327 2023-01-18 22:37

/outputdir/part-r-00000

[cloudera@quickstart Desktop]$ hdfs dfs -cat /outputdir/part-r-00000

'cloudera' 1

'cloudera'. 1

Analyze 1

Data 1

Hadoop 1

Hue 2

In 1

QuickStart 1

Simply 1

VM, 1

Your 1

administrative 1

analyze 1

analyze, 1

and 3

begin 1

data 1

data. 1

easily 1

for 2

in 1

interface 1

is 3

it. 1

lets 1

load 1

open 1

password 1

search, 1

source 1

that 1

the 4

then 1

to 1

username 1

visualize 1

web 1

you 1

your 2

[cloudera@quickstart Desktop]$ hadoop jar /home/cloudera/WordCount.jar

WordCount /inputdir/abc.txt /op1

23/01/18 22:46:23 INFO client.RMProxy: Connecting to ResourceManager

at /0.0.0.0:8032

23/01/18 22:46:23 WARN mapreduce.JobSubmitter: Hadoop command-line

option parsing not performed. Implement the Tool interface and execute

your application with ToolRunner to remedy this. 23/01/18 22:46:24 INFO input.FileInputFormat: Total input paths to process : 1

23/01/18 22:46:24 INFO mapreduce.JobSubmitter: number of splits:1

23/01/18 22:46:24 INFO mapreduce.JobSubmitter: Submitting tokens for

job: job\_1674105093530\_0003

23/01/18 22:46:24 INFO impl.YarnClientImpl: Submitted application

application\_1674105093530\_0003

23/01/18 22:46:25 INFO mapreduce.Job: The url to track the job:

http://quickstart.cloudera:8088/proxy/application\_1674 105093530\_0003/

23/01/18 22:46:25 INFO mapreduce.Job: Running job: job\_1674105093530\_0003

23/01/18 22:46:33 INFO mapreduce.Job: Job job\_1674105093530\_0003

running in uber mode : false

23/01/18 22:46:33 INFO mapreduce.Job: map 0% reduce 0%

23/01/18 22:46:41 INFO mapreduce.Job: map 100% reduce 0%

23/01/18 22:46:49 INFO mapreduce.Job: map 100% reduce 100%

23/01/18 22:46:49 INFO mapreduce.Job: Job job\_1674105093530\_0003

completed successfully

23/01/18 22:46:49 INFO mapreduce.Job: Counters: 49 File System Counters

FILE: Number of bytes read=588 FILE: Number of bytes written=221447 FILE: Number of read operations=0 FILE: Number of large read operations=0 FILE: Number of write operations=0 HDFS: Number of bytes read=402 HDFS: Number of bytes written=327 HDFS: Number of read operations=6 HDFS: Number of large read operations=0 HDFS: Number of write operations=2 Job Counters

Launched map tasks=1

Launched reduce tasks=1

Data-local map tasks=1

Total time spent by all maps in occupied slots (ms)=5627

Total time spent by all reduces in occupied slots (ms)=6162

Total time spent by all map tasks (ms)=5627 Total time spent by all reduce tasks (ms)=6162

Total vcore-seconds taken by all map tasks=5627

Total vcore-seconds taken by all reduce tasks=6162

Total megabyte-seconds taken by all map tasks=5762048

Total megabyte-seconds taken by all reduce tasks=6309888

Map-Reduce Framework

Map input records=3

Map output records=49

Map output bytes=484

Map output materialized bytes=588 Input split bytes=113

Combine input records=0

Combine output records=0

Reduce input groups=39

Reduce shuffle bytes=588

Reduce input records=49

Reduce output records=39

Spilled Records=98

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

GC time elapsed (ms)=143

CPU time spent (ms)=1260

Physical memory (bytes)

snapshot=329596928

Virtual memory (bytes)

snapshot=3007602688

Total committed heap usage

(bytes)=226365440

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=289

File Output Format Counters

Bytes Written=327

[cloudera@quickstart Desktop]$



10) Now we have to move this input file to hdfs. For this we create a direcory on hdfs using command hdfs dfs -mkdir /inputnew.



Then we can verify whether this directory is created or not using ls command **hdfs dfs -ls /**

Move the input file to this directory created in hdfs by using either put command or copyFromLocal command.

Now checking whether the **“abc”** present in /inputdir directory of hdfs or not using **hdfs dfs -ls /inputdir** command

As we can see **“abc”** file is present in /inputdir directory of hdfs. Now we will see the content of this file using **hdfs dfs –cat /inputdir/abc** command



11) Running Mapreduce Program on Hadoop, syntax is hadoop jar jarFileName.jar ClassName /InputFileAddress /outputdir

i.e. **hadoop jar /home/cloudera/WordCount.jar WordCount /inputdir/abc /outputdir Map-Reduce Framework**

****

As we can see in the above output,

**Combine input records=0**

**Combine output records=0**

**We are getting this because we have commented the Combiner line in main function.**

**And Reduce shuffle bytes coming as,**

**Reduce shuffle bytes=1876**

**So when we are not using combiner 1876 bytes acting as an input for the reducer.**

12) Then we can verify the content of outputdir directory and in that part-r file has the actual output by using the command Hdfs dfs -cat /outputdir/part-r-00000 This will give us final output. The same file can also be accessed using a browser. For every execution of this program we need to delete the output directory or give a new name to the output directory every time.

1st we are checking whether the outputdir directory is created in hdfs or not using command **hdfs dfs -ls /**

Now let’s check what we have inside this **outputdir** directory using command as **hdfs dfs -ls /outputdir**

Now we want to read the content of the **part-r-00000** file which present inside the **outputdir** using command **hdfs dfs -cat /outputdir/part-r-00000**

****

**It will give the count of number of times each word has occurred as output. 13) The same file can also be accessed using a browser.**

Browse the Directory by

**Hadoop->HDFS Namenode->Ultilities ->Browse the file system**

Now downloading the **part-r-00000** file**.**

****

Inside the **part-r-00000** file it will have the same output as we are getting after executing using command **hadoop jar /home/cloudera/WordCount.jar WordCount /inputdir/abc /op1**

****

**For every execution of this program we need to delete the output directory or give a new name to the output directory every time.**

**Implementation of WordCount problem using Hadoop MapReduce (With Combiner) in Eclipse:**

1) We will perform the same steps as we have done above for WordCount (without using combiner) in that we just uncommenting the combiner line in main function.



2) And will delete the WordCount.jar file in which all jar files are present from **/home/cloudera.**

****

**We have successfully deleted the WordCount.jar file**

****

3) Now exporting the jar files Right Click on the project name WordCount -> Export -> Java -> JAR File -> Next -> for select the export destination for JAR file: browse -> Name : WordCount.jar -> save in folder -> cloudera -> Finish -> OK





4) Now checking the WordCount.jar file is created or not using **–ls** command



5) Running Mapreduce Program on Hadoop, syntax is hadoop jar jarFileName.jar ClassName /InputFileAddress /outputdir

i.e. **hadoop jar /home/cloudera/WordCount.jar WordCount /inputdir/abc /op1**

here I am using the same input file ‘abc’ which I have created earlier for WordCount example (Without Combiner). **For every execution of this program we need to delete the output directory or give a new name to the output directory every time.** So here I am giving the new name to the output directory as **‘op1’**.



➢ As we can see from above image the the combiner input and output records coming out as, **Combine input records=177**

**Combine output records=84**

Earlier it was coming out as “zero” while executing WordCount (without combiner).

**Combine input records=0**

**Combine output records=0**

➢ And also here we are getting the Reduce Shuffle bytes as,

**Reduce shuffle bytes=942**

Earlier while executing WordCount (without combiner) it is coming out as, **Reduce shuffle bytes=1876**

➢ **So Combiner is used to save the Network Bandwidth. So for saving the Network bandwidth we make use of combiner. So instead of sending every word over the network what we do is we incorporate the logic of the reducer at the combiner side so that the less amount of information can be transmitted over the network.**

➢ **So when we are not using combiner 1876 bytes acting as an input for the reducer. And when we are making use of the combiner so 942 bytes acting as input for the reducer.**

6) The same file can also be accessed using a browser.

Browse the Directory by

**Hadoop->HDFS Namenode->Ultilities ->Browse the file system**

Now downloading the **part-r-00000** file**.**

****

Inside the **part-r-00000** file it will have the same output as we are getting after executing using command **hadoop jar /home/cloudera/WordCount.jar WordCount /inputdir/abc /op1**

****

****Terminal Output -:

[cloudera@quickstart ~]$ ls

cloudera-manager eclipse Music WordCount.jar

cm\_api.py enterprise-deployment.json Pictures workspace Desktop express-deployment.json Public

Documents kerberos Templates

Downloads lib Videos

[cloudera@quickstart ~]$ pwd

/home/cloudera

[cloudera@quickstart ~]$ gedit abc.txt

[cloudera@quickstart ~]$ hdfs dfs -ls /

Found 5 items

drwxr-xr-x - hbase supergroup 0 2023-01-18 21:12 /hbase

drwxr-xr-x - solr solr 0 2015-06-09 03:38 /solr

drwxrwxrwx - hdfs supergroup 0 2023-01-18 21:14 /tmp

drwxr-xr-x - hdfs supergroup 0 2015-06-09 03:38 /user

drwxr-xr-x - hdfs supergroup 0 2015-06-09 03:36 /var

[cloudera@quickstart ~]$ hdfs dfs -mkdir /inputdir

[cloudera@quickstart ~]$ hdfs dfs -ls /

Found 6 items

drwxr-xr-x - hbase supergroup 0 2023-01-18 21:12 /hbase drwxr-xr-x - cloudera supergroup 0 2023-01-18 22:03 /inputdir drwxr-xr-x - solr solr 0 2015-06-09 03:38 /solr

drwxrwxrwx - hdfs supergroup 0 2023-01-18 21:14 /tmp

drwxr-xr-x - hdfs supergroup 0 2015-06-09 03:38 /user

drwxr-xr-x - hdfs supergroup 0 2015-06-09 03:36 /var

[cloudera@quickstart ~]$ hdfs dfs -put

-put: Not enough arguments: expected 1 but got 0

Usage: hadoop fs [generic options] -put [-f] [-p] [-l] <localsrc> ... <dst> [cloudera@quickstart ~]$

~~[cloudera@quickstart ~]$~~

[cloudera@quickstart ~]$

[cloudera@quickstart ~]$

[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/Desktop/abc.txt/inputdir/ put: `/home/cloudera/Desktop/abc.txt/inputdir/': No such file or directory [cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/Desktop/abc.txt /inputdir/ put: `/home/cloudera/Desktop/abc.txt': No such file or directory [cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/Desktop/abc.txt /inputdir/

put: `/home/cloudera/Desktop/abc.txt': No such file or directory [cloudera@quickstart ~]$ ls

abc.txt Desktop eclipse kerberos

Pictures Videos

cloudera-manager Documents enterprise-deployment.json lib Public WordCount.jar

cm\_api.py Downloads express-deployment.json Music Templates workspace

[cloudera@quickstart ~]$ ls Desktop

Eclipse.desktop Enterprise.desktop Express.desktop

[cloudera@quickstart ~]$ cd Desktop

[cloudera@quickstart Desktop]$ gedit abc.txt

[cloudera@quickstart Desktop]$ ls Desktop

ls: cannot access Desktop: No such file or directory

[cloudera@quickstart Desktop]$ ls

abc.txt Eclipse.desktop Enterprise.desktop Express.desktop [cloudera@quickstart Desktop]$ hdfs dfs -put

/home/cloudera/Desktop/abc.txt /inputdir/

[cloudera@quickstart Desktop]$ hdfs dfs -ls /inputdir

Found 1 items

-rw-r--r-- 1 cloudera supergroup 289 2023-01-18 22:21 /inputdir/abc.txt [cloudera@quickstart Desktop]$ hdfs dfs -ls /inputdir

Found 1 items

-rw-r--r-- 1 cloudera supergroup 289 2023-01-18 22:21 /inputdir/abc.txt [cloudera@quickstart Desktop]$ hdfs dfs -cat /inputdir/abc.txt Analyze Your Data

Hue is the open source web interface for Hadoop that lets you analyze your data. Simply load in your data and then easily begin to analyze, search, and visualize it. In the QuickStart VM, the administrative username for Hue is 'cloudera' and the password is 'cloudera'. [cloudera@quickstart Desktop]$ hadoop jar /home/cloudera/WordCount.jar WordCount /inputdir/abc/ /outputdir

23/01/18 22:32:55 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032

23/01/18 22:32:55 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.

23/01/18 22:32:56 INFO mapreduce.JobSubmitter: Cleaning up the staging area /tmp/hadoop-yarn/staging/cloudera/.staging/job\_1674105093530\_0001 23/01/18 22:32:56 WARN security.UserGroupInformation:

PriviledgedActionException as:cloudera (auth:SIMPLE)

~~cause:org.apache.hadoop.mapreduce.lib.input.InvalidInputException:~~

Input path does not exist:

hdfs://quickstart.cloudera:8020/inputdir/abc

Exception in thread "main"

org.apache.hadoop.mapreduce.lib.input.InvalidInputException: Input

path does not exist: hdfs://quickstart.cloudera:8020/inputdir/abc

at

org.apache.hadoop.mapreduce.lib.input.FileInputFormat.singleThreadedListStatus(FileInput Format.java:321)

at

org.apache.hadoop.mapreduce.lib.input.FileInputFormat.listStatus(FileInputFormat.java:264 )

at

org.apache.hadoop.mapreduce.lib.input.FileInputFormat.getSplits(FileInputFormat.java:385) at org.apache.hadoop.mapreduce.JobSubmitter.writeNewSplits(JobSubmitter.java:597) at org.apache.hadoop.mapreduce.JobSubmitter.writeSplits(JobSubmitter.java:614) at

org.apache.hadoop.mapreduce.JobSubmitter.submitJobInternal(JobSubmitter.java:492) at org.apache.hadoop.mapreduce.Job$10.run(Job.java:1306)

at org.apache.hadoop.mapreduce.Job$10.run(Job.java:1303)

at java.security.AccessController.doPrivileged(Native Method)

at javax.security.auth.Subject.doAs(Subject.java:415)

at

org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1671) at org.apache.hadoop.mapreduce.Job.submit(Job.java:1303)

at org.apache.hadoop.mapreduce.Job.waitForCompletion(Job.java:1324) at WordCount.main(WordCount.java:59)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57) at

sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) at java.lang.reflect.Method.invoke(Method.java:606)

at org.apache.hadoop.util.RunJar.run(RunJar.java:221)

at org.apache.hadoop.util.RunJar.main(RunJar.java:136)

[cloudera@quickstart Desktop]$ hadoop jar /home/cloudera/WordCount.jar WordCount /inputdir/abc.txt /outputdir

23/01/18 22:36:40 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032

23/01/18 22:36:41 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.

23/01/18 22:36:41 INFO input.FileInputFormat: Total input paths to process : 1 23/01/18 22:36:41 INFO mapreduce.JobSubmitter: number of splits:1

23/01/18 22:36:42 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1674105093530\_0002

23/01/18 22:36:42 INFO impl.YarnClientImpl: Submitted application

application\_1674105093530\_0002

23/01/18 22:36:42 INFO mapreduce.Job: The url to track the job:

http://quickstart.cloudera:8088/proxy/application\_1674105093530\_0002/ 23/01/18 22:36:42 INFO mapreduce.Job: Running job: job\_1674105093530\_0002 23/01/18 22:36:55 INFO mapreduce.Job: Job job\_1674105093530\_0002 ~~running in uber mode : false~~

23/01/18 22:36:55 INFO mapreduce.Job: map 0% reduce 0% 23/01/18 22:37:05 INFO mapreduce.Job: map 100% reduce 0% 23/01/18 22:37:15 INFO mapreduce.Job: map 100% reduce 100% 23/01/18 22:37:15 INFO mapreduce.Job: Job job\_1674105093530\_0002 completed successfully

23/01/18 22:37:15 INFO mapreduce.Job: Counters: 49

File System Counters

FILE: Number of bytes read=588

FILE: Number of bytes written=221459

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=402

HDFS: Number of bytes written=327

HDFS: Number of read operations=6

HDFS: Number of large read operations=0

HDFS: Number of write operations=2

Job Counters

Launched map tasks=1

Launched reduce tasks=1

Data-local map tasks=1

Total time spent by all maps in occupied slots (ms)=8456 Total time spent by all reduces in occupied slots (ms)=7017 Total time spent by all map tasks (ms)=8456

Total time spent by all reduce tasks (ms)=7017 Total vcore-seconds taken by all map tasks=8456 Total vcore-seconds taken by all reduce tasks=7017 Total megabyte-seconds taken by all map tasks=8658944 Total megabyte-seconds taken by all reduce tasks=7185408 Map-Reduce Framework

Map input records=3

Map output records=49

Map output bytes=484

Map output materialized bytes=588

Input split bytes=113

Combine input records=0

Combine output records=0

Reduce input groups=39

Reduce shuffle bytes=588

Reduce input records=49

Reduce output records=39

Spilled Records=98

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

GC time elapsed (ms)=213

CPU time spent (ms)=1330

Physical memory (bytes) snapshot=342401024 Virtual memory (bytes) snapshot=3007152128 Total committed heap usage (bytes)=226365440  ~~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=289

File Output Format Counters

Bytes Written=327

[cloudera@quickstart Desktop]$ hdfs dfs -ls /outputdir Found 2 items

-rw-r--r-- 1 cloudera supergroup 0 2023-01-18 22:37 /outputdir/\_SUCCESS

-rw-r--r-- 1 cloudera supergroup 327 2023-01-18 22:37 /outputdir/part-r-00000

[cloudera@quickstart Desktop]$ hdfs dfs -cat /outputdir/part-r-00000 'cloudera' 1

'cloudera'. 1

Analyze 1

Data 1

Hadoop 1

Hue 2

In 1

QuickStart 1

Simply 1

VM, 1

Your 1

administrative 1

analyze 1

analyze, 1

and 3

begin 1

data 1

data. 1

easily 1

for 2

in 1

interface 1

is 3

it. 1

lets 1

load 1

open 1

password 1

search, 1

source 1

that 1

the 4

then 1

~~to 1~~

username 1

visualize 1

web 1

you 1

your 2

[cloudera@quickstart Desktop]$ hadoop jar /home/cloudera/WordCount.jar WordCount /inputdir/abc.txt /op1

23/01/18 22:46:23 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032

23/01/18 22:46:23 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.

23/01/18 22:46:24 INFO input.FileInputFormat: Total input paths to process : 1 23/01/18 22:46:24 INFO mapreduce.JobSubmitter: number of splits:1 23/01/18 22:46:24 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1674105093530\_0003

23/01/18 22:46:24 INFO impl.YarnClientImpl: Submitted application application\_1674105093530\_0003

23/01/18 22:46:25 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application\_1674105093530\_0003/ 23/01/18 22:46:25 INFO mapreduce.Job: Running job: job\_1674105093530\_0003 23/01/18 22:46:33 INFO mapreduce.Job: Job job\_1674105093530\_0003 running in uber mode : false

23/01/18 22:46:33 INFO mapreduce.Job: map 0% reduce 0% 23/01/18 22:46:41 INFO mapreduce.Job: map 100% reduce 0% 23/01/18 22:46:49 INFO mapreduce.Job: map 100% reduce 100% 23/01/18 22:46:49 INFO mapreduce.Job: Job job\_1674105093530\_0003 completed successfully

23/01/18 22:46:49 INFO mapreduce.Job: Counters: 49

File System Counters

FILE: Number of bytes read=588

FILE: Number of bytes written=221447

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=402

HDFS: Number of bytes written=327

HDFS: Number of read operations=6

HDFS: Number of large read operations=0

HDFS: Number of write operations=2

Job Counters

Launched map tasks=1

Launched reduce tasks=1

Data-local map tasks=1

Total time spent by all maps in occupied slots (ms)=5627 Total time spent by all reduces in occupied slots (ms)=6162 Total time spent by all map tasks (ms)=5627

Total time spent by all reduce tasks (ms)=6162

Total vcore-seconds taken by all map tasks=5627

Total vcore-seconds taken by all reduce tasks=6162  ~~Total megabyte-seconds taken by all map tasks=5762048~~

Total megabyte-seconds taken by all reduce tasks=6309888 Map-Reduce Framework

Map input records=3

Map output records=49

Map output bytes=484

Map output materialized bytes=588

Input split bytes=113

Combine input records=0

Combine output records=0

Reduce input groups=39

Reduce shuffle bytes=588

Reduce input records=49

Reduce output records=39

Spilled Records=98

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

GC time elapsed (ms)=143

CPU time spent (ms)=1260

Physical memory (bytes) snapshot=329596928 Virtual memory (bytes) snapshot=3007602688 Total committed heap usage (bytes)=226365440 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=289

File Output Format Counters

Bytes Written=327

[cloudera@quickstart Desktop]$