**Vivekanand Education Society’s Institute of Technology**

**Department of Computer Engineering**



A Report on

**Phrase Count Using Apache Hadoop on Cloudera**

**PROJECT MENTOR**

Mrs. Sujata Khedkar  
Mrs. Asha Bharambe  
Mrs. Sangeeta Oswal

**GROUP MEMBERS**

Aakash Goplani (D17A-29)   
Chirag Hingorani (D17A-36)   
Rupesh Mishra (D17A-60)

Academic Year 2014-2015

**Vivekanand Education Society’s Institute of Technology**

**Department of Computer Engineering**



**CERTIFICATE**

**Phrase Count Using Apache Hadoop on Cloudera**

**PROJECT MENTOR**

Mrs. Sujata Khedkar  
Mrs. Asha Bharambe  
Mrs. Sangeeta Oswal

**SUBMITTED BY**

Aakash Goplani (D17A-29)   
Chirag Hingorani (D17A-36)   
Rupesh Mishra (D17A-60)

Academic Year 2014-2015

\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

Date Examiner

\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

Principal HOD

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **TOPIC** | **Pg. No.** |
| **1** | **Chapter 1 – Introduction** | 4 |
| 1.1 | Problem Statement | 4 |
|  | | |
| **2** | **Chapter 2 – Requirements** | 5 |
| 2.1 | Functional Requirements | 5 |
| 2.2 | Non-Functional Requirements | 5 |
| 2.3 | Hardware & Software Requirements | 5 |
|  | | |
| **3** | **Chapter 3 – Design of the project** | 6 |
| 3.1 | Flowchart | 6 |
|  | | |
| **4** | **Chapter 4 – Implementation** | 7 |
| 4.1 | Code | 7 |
| 4.2 | Output / Results | 10 |
|  | | |
| **5** | **Chapter 5 – Conclusion** | 13 |
| 5.1 | Conclusion | 13 |
| 5.2 | Acknowledgement | 13 |

**1. INTRODUCTION**

**1.1 PROBLEM STATEMENT**

Read unstructured text data and find the following:

For a given phrase length of "m" words (where 1 <= m <= 4) and a given number "n", find the top "n" recurring phases of length "m" in the given text and the corresponding frequency count.

**2. REQUIREMENTS**

**2.1 FUNCTIONAL REQUIREMENTS**

* Should be able to find the top "n" recurring phases of length "m" in the given text.
* Display the frequency count of each recurring phrase.

**2.2 NON-FUNCTIONAL REQUIREMENTS**

* Precision – The system should be precise enough to scan the entire document and find the desired phrases of a given length.
* Efficient – Total processing time and storage space taken by system should be minimal.
* Scalable – The system should be able to scale-up and scale-down as per the length of the input text file, generation of phrases etc.

**2.3 HARDWARE REQUIREMENTS**

* Centos
* JDK 1.8.0\_31

**3. DESIGN OF PROJECT**

**3.1 FLOWCHART**

START

TAKE INPUT FILE FROM USER

MAPPER

KEY-VALUE PAIR GENERATION

SHUFFLE-SORT

REDUCER

OUTPUT

END

**4. IMPLEMENTATION**

**4.1 CODE**

**Phrase Driver:**

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

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

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

import org.apache.hadoop.mapreduce.Job;

public class PhraseDriver

{

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

{

if (args.length != 2)

{

System.out.printf("Usage: PhraseCount <input dir> <output dir>\n");

System.exit(-1);

}

Job job = new Job();

job.setJarByClass(PhraseDriver.class);

job.setJobName("Phrase Count");

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

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

job.setMapperClass(PhraseMapper.class);

job.setNumReduceTasks(0);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

boolean success = job.waitForCompletion(true);

System.exit(success ? 0 : 1);

}

}

**Phrase Map: Mapper**

import java.io.IOException;

import java.util.\*;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class PhraseMapper extends Mapper<LongWritable, Text, Text, IntWritable>

{

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

{

String input = value.toString();

input = input.toLowerCase();

input = input.replaceAll("[\",!.:;?\*()']", "");

input = input.replaceAll("-", "");

String[] split = input.split(" ");

Map<String, Integer> counts = new HashMap<String,Integer>(split.length\*(split.length-1)/2,1.0f);

int idx0 = 0;

for(int i=0; i<split.length-1; i++)

{

int splitIpos = input.indexOf(split[i],idx0);

int newPhraseLen = splitIpos - idx0 + split[i].length();

String phrase = input.substring(idx0, idx0 + newPhraseLen);

for(int j=i+1; j<split.length; j++)

{

newPhraseLen = phrase.length()+split[j].length()+1;

phrase=input.substring(idx0, idx0 + newPhraseLen);

Integer count = counts.get(phrase);

if(count==null)

{

counts.put(phrase, 1);

}

else

{

counts.put(phrase, count+1);

}

}

idx0 = splitIpos+split[i].length()+1;

}

Map.Entry<String, Integer>[] entries = counts.entrySet().toArray(new Map.Entry[0]);

Arrays.sort(entries, new Comparator<Map.Entry<String, Integer>>()

{

@Override

public int compare(Map.Entry<String, Integer> o1, Map.Entry<String, Integer> o2)

{

return o2.getValue().compareTo(o1.getValue());

}

});

for(Map.Entry<String,Integer> entry:entries)

{

int count = entry.getValue();

String keyans = entry.getKey();

if(count>1)

{

context.write(new Text(keyans), new IntWritable(count));

}

}

}

}

**Phrase Red: Reducer**

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class PhraseReducer extends Reducer<Text, IntWritable, Text, IntWritable>

{

@Override

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

{

int phraseCount = 0;

while (values.iterator().hasNext())

{

values.iterator().next();

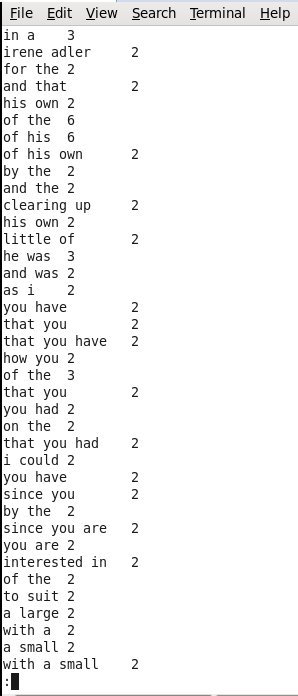
phraseCount++;

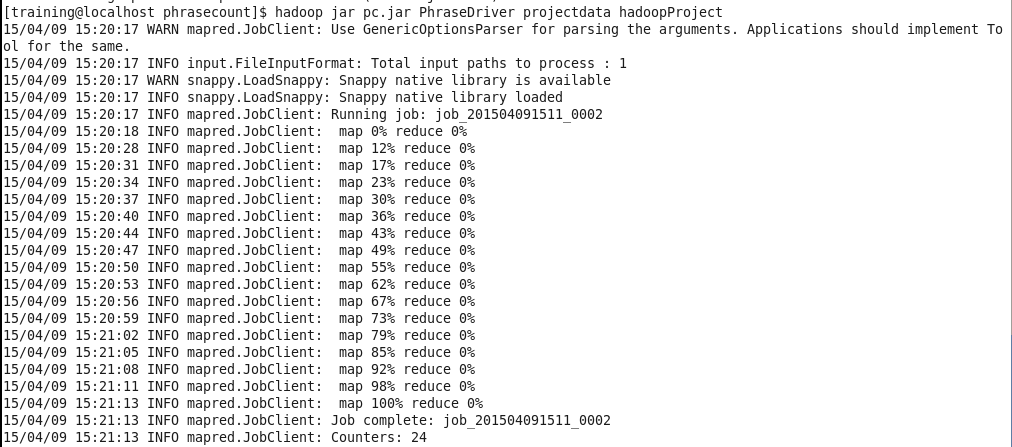
}

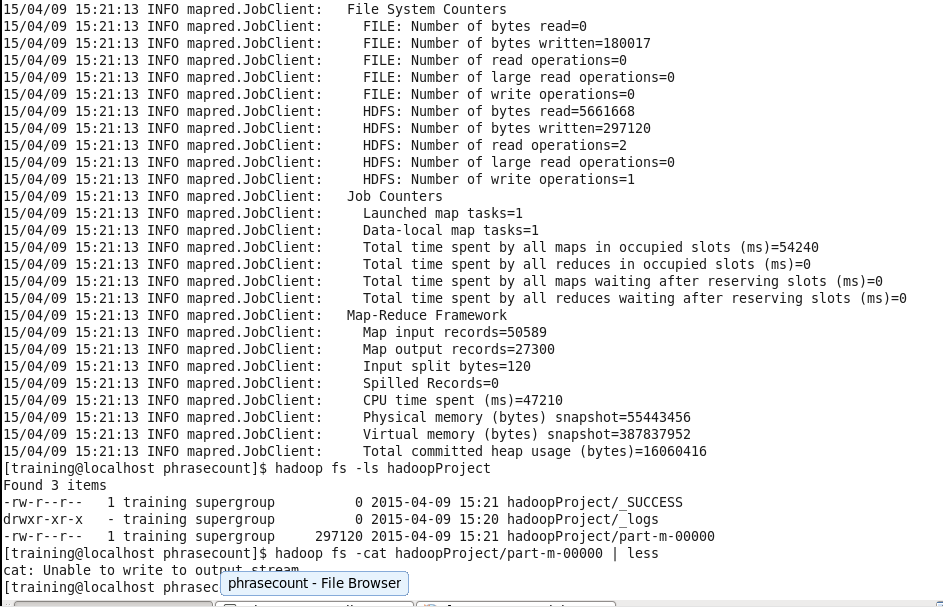
context.write(key, new IntWritable(phraseCount));

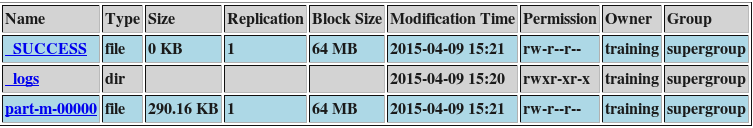
}

**4.2 OUTPUT/RESULTS**









snapshot4.PNG

**5. CONCLUSION**

**5.1 CONCLUSION**

The project involved extracting phrases of length “m” and finding top “n” recurring phrases along with their frequency count using Apache Hadoop on Cloudera.

**We realized the following advantages of Hadoop-**

1) Distributed data and computation.

2) Tasks are independent. So-

* We can easy to handle partial failure. Here the entire nodes can fail and restart.
* It avoids crawling horrors of failure and tolerant synchronous distributed systems.

3) Simple programming model. The end-user programmer only writes map-reduce tasks.

4) Flat scalability.

The requirements of the project were satisfied and implemented successfully.

**5.2 ACKNOWLEDGEMENT**

We are grateful to **Mrs. Sujata Khedkar, Mrs. Asha Bharambe and Mrs. Sangeeta Oswal** for giving us this opportunity to learn Hadoop. We thank **Cloudera** and **VESIT** for organizing this co-curricular course to explore Hadoop.