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

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

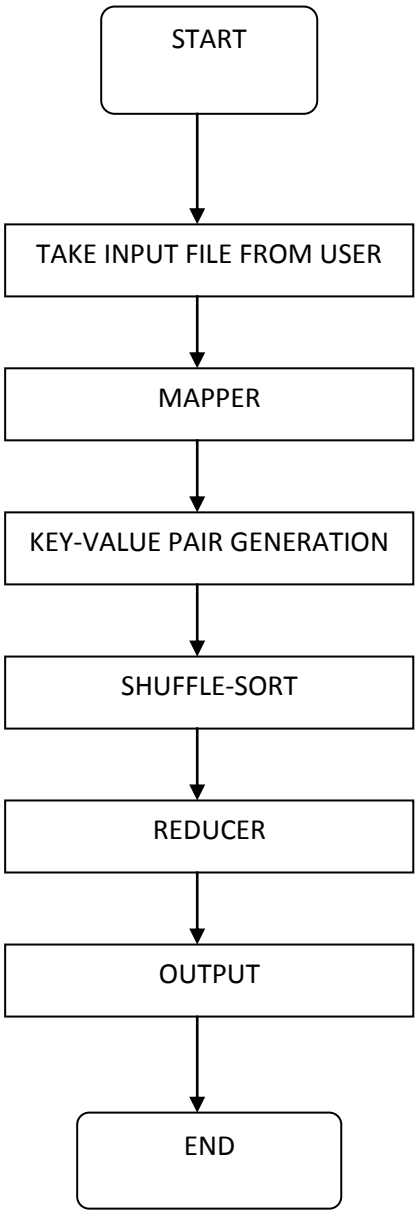
- Precision – The system should be precise enough to scan the entire document and find the desired phrases of a given length.
- Efficient – Total processing time and storage space taken by system should be minimal.
- Scalable – The system should be able to scale-up and scale-down as per the length of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



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

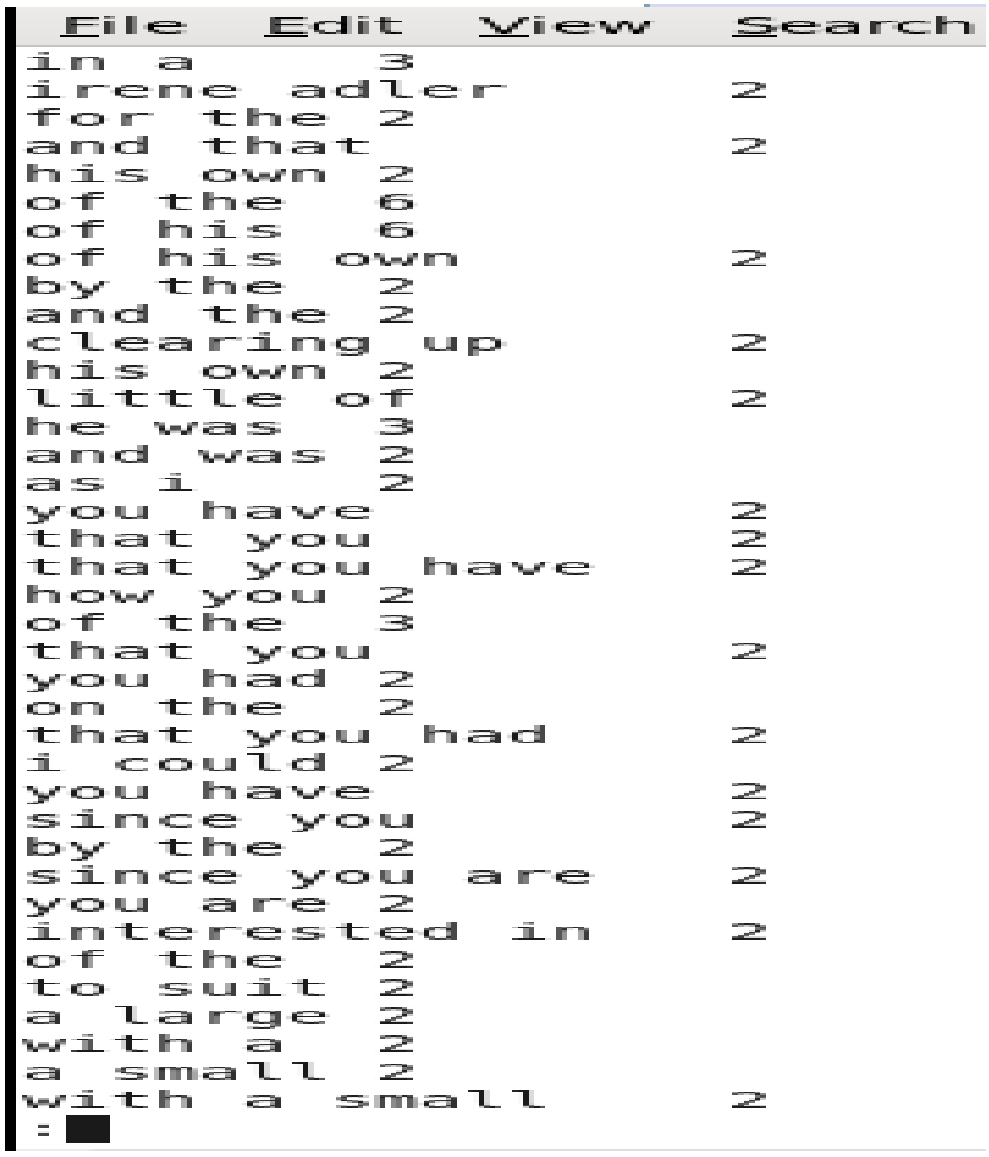
```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

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



```
[training@localhost phrasecount]$ hadoop jar pc.jar PhraseDriver projectdata hadoopProject
15/04/09 15:20:17 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement To
ol for the same.
15/04/09 15:20:17 INFO input.FileInputFormat: Total input paths to process : 1
15/04/09 15:20:17 WARN snappy.LoadSnappy: Snappy native library is available
15/04/09 15:20:17 INFO snappy.LoadSnappy: Snappy native library loaded
15/04/09 15:20:17 INFO mapred.JobClient: Running job: job_201504091511_0002
15/04/09 15:20:18 INFO mapred.JobClient:  map 0% reduce 0%
15/04/09 15:20:28 INFO mapred.JobClient:  map 12% reduce 0%
15/04/09 15:20:31 INFO mapred.JobClient:  map 17% reduce 0%
15/04/09 15:20:34 INFO mapred.JobClient:  map 23% reduce 0%
15/04/09 15:20:37 INFO mapred.JobClient:  map 30% reduce 0%
15/04/09 15:20:40 INFO mapred.JobClient:  map 36% reduce 0%
15/04/09 15:20:44 INFO mapred.JobClient:  map 43% reduce 0%
15/04/09 15:20:47 INFO mapred.JobClient:  map 49% reduce 0%
15/04/09 15:20:50 INFO mapred.JobClient:  map 55% reduce 0%
15/04/09 15:20:53 INFO mapred.JobClient:  map 62% reduce 0%
15/04/09 15:20:56 INFO mapred.JobClient:  map 67% reduce 0%
15/04/09 15:20:59 INFO mapred.JobClient:  map 73% reduce 0%
15/04/09 15:21:02 INFO mapred.JobClient:  map 79% reduce 0%
15/04/09 15:21:05 INFO mapred.JobClient:  map 85% reduce 0%
15/04/09 15:21:08 INFO mapred.JobClient:  map 92% reduce 0%
15/04/09 15:21:11 INFO mapred.JobClient:  map 98% reduce 0%
15/04/09 15:21:13 INFO mapred.JobClient:  map 100% reduce 0%
15/04/09 15:21:13 INFO mapred.JobClient: Job complete: job_201504091511_0002
15/04/09 15:21:13 INFO mapred.JobClient: Counters: 24
```

```
15/04/09 15:21:13 INFO mapred.JobClient:   File System Counters
15/04/09 15:21:13 INFO mapred.JobClient:     FILE: Number of bytes read=0
15/04/09 15:21:13 INFO mapred.JobClient:     FILE: Number of bytes written=180017
15/04/09 15:21:13 INFO mapred.JobClient:     FILE: Number of read operations=0
15/04/09 15:21:13 INFO mapred.JobClient:     FILE: Number of large read operations=0
15/04/09 15:21:13 INFO mapred.JobClient:     FILE: Number of write operations=0
15/04/09 15:21:13 INFO mapred.JobClient:     HDFS: Number of bytes read=5661668
15/04/09 15:21:13 INFO mapred.JobClient:     HDFS: Number of bytes written=297120
15/04/09 15:21:13 INFO mapred.JobClient:     HDFS: Number of read operations=2
15/04/09 15:21:13 INFO mapred.JobClient:     HDFS: Number of large read operations=0
15/04/09 15:21:13 INFO mapred.JobClient:     HDFS: Number of write operations=1
15/04/09 15:21:13 INFO mapred.JobClient:   Job Counters
15/04/09 15:21:13 INFO mapred.JobClient:     Launched map tasks=1
15/04/09 15:21:13 INFO mapred.JobClient:     Data-local map tasks=1
15/04/09 15:21:13 INFO mapred.JobClient:     Total time spent by all maps in occupied slots (ms)=54240
15/04/09 15:21:13 INFO mapred.JobClient:     Total time spent by all reduces in occupied slots (ms)=0
15/04/09 15:21:13 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0
15/04/09 15:21:13 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
15/04/09 15:21:13 INFO mapred.JobClient:   Map-Reduce Framework
15/04/09 15:21:13 INFO mapred.JobClient:     Map input records=50589
15/04/09 15:21:13 INFO mapred.JobClient:     Map output records=27300
15/04/09 15:21:13 INFO mapred.JobClient:     Input split bytes=120
15/04/09 15:21:13 INFO mapred.JobClient:     Spilled Records=0
15/04/09 15:21:13 INFO mapred.JobClient:     CPU time spent (ms)=47210
15/04/09 15:21:13 INFO mapred.JobClient:     Physical memory (bytes) snapshot=55443456
15/04/09 15:21:13 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=387837952
15/04/09 15:21:13 INFO mapred.JobClient:     Total committed heap usage (bytes)=16060416
[training@localhost phrasecount]$ hadoop fs -ls hadoopProject
Found 3 items
-rw-r--r--   1 training supergroup          0 2015-04-09 15:21 hadoopProject/_SUCCESS
drwxr-xr-x   - training supergroup          0 2015-04-09 15:20 hadoopProject/_logs
-rw-r--r--   1 training supergroup     297120 2015-04-09 15:21 hadoopProject/part-m-00000
[training@localhost phrasecount]$ hadoop fs -cat hadoopProject/part-m-00000 | less
cat: Unable to write to output stream.
[training@localhost phrasec
```

phrasecount - File Browser

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_SUCCESS	file	0 KB	1	64 MB	2015-04-09 15:21	rw-r--r--	training	supergroup
_logs	dir				2015-04-09 15:20	rwxr-xr-x	training	supergroup
part-m-00000	file	290.16 KB	1	64 MB	2015-04-09 15:21	rw-r--r--	training	supergroup

hadoopProject	dir				2015-04-09 15:21	rwxr-xr-x	training	supergroup

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.