BDPA - Assignment 1

Hippolyte JACOMET - github.com/hippolytej/BDPA-HW

1 HADOOP SETUP

Here's my Hadoop setup, didn't change the defaults.

```
[cloudera@quickstart ~]$ hadoop version
Hadoop 2.6.0-cdh5.8.0
Subversion http://github.com/cloudera/hadoop -r
57e7b8556919574d517e874abfb7ebe31a366c2b
Compiled by jenkins on 2016-06-16T19:38Z
Compiled with protoc 2.5.0
From source with checksum 9e99ecd28376acfd5f78c325dd939fed
This command was run using /usr/lib/hadoop/hadoop-common-2.6.0-cdh5.8.0.jar
[cloudera@quickstart ~]$ hadoop checknative -a
17/02/16 11:55:57 INFO bzip2.Bzip2Factory: Successfully loaded &
initialized native-bzip2 library system-native
17/02/16 11:55:57 INFO zlib.ZlibFactory: Successfully loaded & initialized
native-zlib library
Native library checking:
hadoop:  true /usr/lib/hadoop/lib/native/libhadoop.so.1.0.0
zlib:    true /lib64/libz.so.1
snappy:  true /usr/lib/hadoop/lib/native/libsnappy.so.1
lz4:     true revision:10301
bzip2:   true /lib64/libbz2.so.1
openssl: true /usr/lib64/libcrypto.so
```

As for my virtual machine:

- Memory 4096 Mo;
- 1 Cores 100% allocated (later added one for inverted index, was way too slow);
- 24 Mo of Graphics Memory (default was also too light).

2 IMPLEMENTATIONS

2.1 STOP WORDS

2.1.1 Main

Based on the wordcount main of the tutorial found at wiki.apache.org.

```
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();

        Job job = new Job(conf, "StopWords");
        job.setJarByClass(StopWords.class);


        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        job.setMapperClass(Map.class);
//      job.setCombinerClass(Combiner.class);
        job.setReducerClass(Reduce.class);
        job.setNumReduceTasks(10);
//      job.setNumReduceTasks(50);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.setInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

//      conf.set("mapreduce.map.output.compress", "true");
//      conf.set("mapreduce.map.output.compress.codec", "org.apache.ha-
doop.io.compress.SnappyCodec");

        job.waitForCompletion(true);
    }
```

2.1.2 Mapper

Didn't change much the code used in the wordcount tutorial, mainly personalized the tokenizer to avoid inappropriate "words":

```
    public static class Map extends Mapper<LongWritable, Text, Text,
IntWritable> {
        private final static IntWritable ONE = new IntWritable(1);
        private Text word = new Text();

        @Override
        public void map(LongWritable key, Text value, Context context)
                throws IOException, InterruptedException {
            String line = value.toString().toLowerCase();
            StringTokenizer tokenizer = new StringTokenizer(line, "
\t\n\r\f,.:;?![]{}'\"()&<>~_-12345677890#$*^%/@\\`=+|");
            while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
                context.write(word, ONE);
            }
        }
    }
```

2.1.3 Combiner

Had to create a custom combiner as I could not simply use the reducer class because I chose to write only the words and not the count in the reducer (cf. next section)

```
    public static class Combiner extends Reducer<Text,IntWritable,
Text,IntWritable> {
        @Override
        public void reduce(Text key, Iterable<IntWritable> values, Context
context) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            context.write(key, new IntWritable(sum));
        }
    }
```

2.1.4 Reducer

Simply added an if condition to the wordcount reducer, but also chose to not write the count which is useless: I changed the output to a NullWritable.

```
    public static class Reduce extends Reducer<Text, IntWritable, Text,
NullWritable> {
        @Override
        public void reduce(Text key, Iterable<IntWritable> values, Context
context)
                throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            if (sum > 4000){
                context.write(key, NullWritable.get());
            }
        }
    }
```

2.2 INVERTED INDEX

I detail here only the code of the inverted index with frequencies, which is more complete and features the counter. Code for the basic inverted index one can be found in my GitHub repository.

2.2.1 Main & counter

The main is quite similar to the one from stopwords apart from the counter which is saved in a file on hdfs.

```
    static enum CustomCounters {UNIQUEWORDS}
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();

        Job job = new Job(conf, "InvertedIndex");
        job.setJarByClass(InvertedIndex.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        job.setMapperClass(Map.class);
        job.setCombinerClass(Combiner.class);
        job.setReducerClass(Reduce.class);
        job.setNumReduceTasks(10);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.setInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        conf.set("mapreduce.map.output.compress", "true");
        conf.set("mapreduce.map.output.compress.codec", "org.apache.ha-
doop.io.compress.SnappyCodec");


        job.waitForCompletion(true);
        Counter counter = job.getCounters().findCounter(Cus-
tomCounters.UNIQUEWORDS);

        FileSystem hdfs = FileSystem.get(URI.create("count"), conf);
        Path file = new Path("counter.txt");
        if ( hdfs.exists( file )) { hdfs.delete( file, true ); }
        OutputStream os = hdfs.create(file);
        BufferedWriter br = new BufferedWriter( new OutputStreamWriter( os,
"UTF-8" ) );
        br.write("Unique words in a single file = " + counter.getValue());
        br.close();
        hdfs.close();
    }
```

2.2.2 Mapper

Based on the same mapper as the one used for Stopwords, only changed the values output to Text, and wrote the file names extracted from the context variable.

```
    public static class Map extends Mapper<LongWritable, Text, Text, Text>
{
        private Text word = new Text();

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

            FileSplit split = (FileSplit) context.getInputSplit();
            String filename = split.getPath().getName().toString();

            String line = value.toString().toLowerCase();
            StringTokenizer tokenizer = new StringTokenizer(line, "
\t\n\r\f,.:;?![]{}'\"()&<>~_-12345677890#$*^%/@\\`=+|");
            while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
                context.write(word, new Text(filename));
            }
        }
    }
```

2.2.3 Combiner

Had to create a custom combiner which slightly differs from the reducer. A hashmap is created as an intermediate index, saved as suggested in the form of doc1#count1,doc2#count2,… so as to later be easily processed by the reducers.

```
    public static class Combiner extends Reducer<Text,Text, Text,Text> {
        @Override
        public void reduce(Text key, Iterable<Text> values, Context con-
text) throws IOException, InterruptedException {
            HashMap<String, Integer> countDooku = new HashMap<String, Inte-
ger>();

            for (Text val : values) {
                if(countDooku.containsKey(val.toString())){
                    countDooku.put(val.toString(), count-
Dooku.get(val.toString()) + 1);
                }else{
                    countDooku.put(val.toString(), 1);
                }
            }

            String filesFrequency = new String();
            for (String fileName: countDooku.keySet()){
                String freq = countDooku.get(fileName).toString();
                filesFrequency = filesFrequency + fileName + "#" + freq +
",";
            }

            filesFrequency =  filesFrequency.substring(0, filesFre-
quency.length()-1);

            Text index = new Text();
            index.set(filesFrequency.toString());
            context.write(key, index);
        }
    }
```

2.2.4 Reducer

I chose to rule out the stop words only in the reducer as it is where the number of comparison with stopwords.csv would be minimal as the keys have been concatenated. Hence, the reducer starts with importing the .csv file, and checking if the key is a stop word. I used code for both local and hdfs files to be able to debug in standalone mode.

In the reduce method, a hashmap is used in the same fashion as the combiner, except the input is slightly different and must be split along “,” and “#”.

Finally, the counter is incremented if a key has only one document name in its output value, that is, simply if no comma can be found in it.

```
    public static class Reduce extends Reducer<Text, Text, Text, Text> {
        String stopwords = new String();
        public void setup(Context context) throws IOException,
InterruptedException {
//      Check if key is a stop word ----------------------
//          Test with local file for standalone mode
//          File file = new File("stopwords.csv");
//          Scanner sw = new Scanner(file);
//      With DHFS file
            Path pt=new Path("stopwords.csv");
            FileSystem fs = FileSystem.get(new Configuration());
            Scanner sw=new Scanner(fs.open(pt));

            while (sw.hasNext()){
                stopwords = stopwords + " " + sw.next().toString();
            }
            sw.close();
        }
        @Override
        public void reduce(Text key, Iterable<Text> values, Context
context)
                throws IOException, InterruptedException {
//          Create index and count frequencies ---------------
            if (!stopwords.contains(key.toString())){
                HashMap<String, Integer> countDooku = new HashMap<String,
Integer>();
                for (Text val : values) {
                    for (String token: val.toString().split(",")) {
                        String[] couple = token.split("#");
                        if(countDooku.containsKey(couple[0])){
                            countDooku.put(couple[0],
countDooku.get(couple[0]) + Integer.parseInt(couple[1]));
                        }
                        else{
                            countDooku.put(couple[0],
Integer.parseInt(couple[1]));
                        }
                    }
                }
                String finalFilesFrequency = new String();
                for (String fileName: countDooku.keySet()){
                    String freq = countDooku.get(fileName).toString();
                    finalFilesFrequency = finalFilesFrequency + fileName +
"#" + freq + ", ";
                }
                finalFilesFrequency =  finalFilesFrequency.substring(0,
finalFilesFrequency.length()-2);
                Text finalIndex = new Text();
                finalIndex.set(finalFilesFrequency);
                context.write(key, finalIndex);
//              Here comes the counter...
                if (!finalFilesFrequency.contains(",")){

context.getCounter(CustomCounters.UNIQUEWORDS).increment(1);
                }
            }
        }
    }
```

3 DATA SET

I made the simplification of excluding all numbers and not count them as words, even though there may be dates, and more generally many special characters that are less controversial.

4 TEST SCENARIOS

The procedure for running tests was the following: update code on eclipse accordingly to test (adjust the number of reducers…), export as jar, and run with Hadoop.
Results were concatenated and saved on hdfs using:

```
hadoop fs -cat output/part\* | hadoop fs -put - name.text
```

And then locally downloaded with:

```
Hadoop -get name.text
```

4.1 STOPWORDS WITH 10 REDUCERS

Execution time : 2'19''

Job Overview

Job Name:	StopWords
User Name:	cloudera
Queue:	root.cloudera
State:	SUCCEEDED
Uberized:	false
Submitted:	Wed Feb 15 06:31:29 PST 2017
Started:	Wed Feb 15 06:31:38 PST 2017
Finished:	Wed Feb 15 06:33:58 PST 2017
Elapsed:	2mins, 19sec
Diagnostics:	
Average Map Time	35sec
Average Shuffle Time	36sec
Average Merge Time	5sec
Average Reduce Time	9sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Wed Feb 15 06:31:32 PST 2017	quickstart.cloudera:8042	logs

Task Type	Total	Complete
Map	3	3
Reduce	10	10

Attempt Type	Failed	Killed	Successful
Maps	0	0	3
Reduces	0	3	10

```
17/02/15 06:34:00 INFO mapreduce.Job: Job job_1485279029005_0017 completed
successfully
17/02/15 06:34:00 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=51484994
                FILE: Number of bytes written=104487032
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=26058246
                HDFS: Number of bytes written=611
                HDFS: Number of read operations=39
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=20
        Job Counters
                Killed reduce tasks=3
                Launched map tasks=3
                Launched reduce tasks=13
                Data-local map tasks=3
                Total time spent by all maps in occupied slots (ms)=105366
                Total time spent by all reduces in occupied slots
(ms)=551481
                Total time spent by all map tasks (ms)=105366
                Total time spent by all reduce tasks (ms)=551481
                Total vcore-seconds taken by all map tasks=105366
                Total vcore-seconds taken by all reduce tasks=551481
                Total megabyte-seconds taken by all map tasks=107894784
                Total megabyte-seconds taken by all reduce tasks=564716544
        Map-Reduce Framework
                Map input records=507535
                Map output records=4603237
                Map output bytes=42278460
                Map output materialized bytes=51485114
                Input split bytes=381
                Combine input records=0
                Combine output records=0
                Reduce input groups=56726
                Reduce shuffle bytes=51485114
                Reduce input records=4603237
                Reduce output records=139
                Spilled Records=9206474
                Shuffled Maps =30
                Failed Shuffles=0
                Merged Map outputs=30
                GC time elapsed (ms)=3699
                CPU time spent (ms)=41830
                Physical memory (bytes) snapshot=1748418560
                Virtual memory (bytes) snapshot=19566936064
                Total committed heap usage (bytes)=1104359424
        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=26057865
        File Output Format Counters
                Bytes Written=611
```

4.2 STOPWORDS WITH 10 REDUCERS AND COMBINER

Execution time : 1’27’’ A significant improvement is observed here, which is what we expect of the combiner : lighter files are sent to the reducer.

Job Overview	
Job Name:	StopWords
User Name:	cloudera
Queue:	root.cloudera
State:	SUCCEEDED
Uberized:	false
Submitted:	Thu Feb 16 14:47:15 PST 2017
Started:	Thu Feb 16 14:47:23 PST 2017
Finished:	Thu Feb 16 14:48:42 PST 2017
Elapsed:	1mins, 19sec
Diagnostics:	
Average Map Time	29sec
Average Shuffle Time	19sec
Average Merge Time	0sec
Average Reduce Time	2sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Thu Feb 16 14:47:18 PST 2017	quickstart.cloudera:8042	logs

Task Type	Total	Complete
Map	3	3
Reduce	10	10

Attempt Type	Failed	Killed	Successful
Maps	0	0	3
Reduces	0	0	10

```
17/02/16 14:48:44 INFO mapreduce.Job: Job job_1487258453206_0005 completed
successfully
17/02/16 14:48:44 INFO mapreduce.Job: Counters: 51
    File System Counters
        FILE: Number of bytes read=1240085
        FILE: Number of bytes written=3999308
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=26058246
        HDFS: Number of bytes written=611
        HDFS: Number of read operations=39
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=20
    Job Counters
        Killed map tasks=1
        Killed reduce tasks=2
        Launched map tasks=3
        Launched reduce tasks=10
        Data-local map tasks=3
        Total time spent by all maps in occupied slots (ms)=89028
        Total time spent by all reduces in occupied slots (ms)=228144
        Total time spent by all map tasks (ms)=89028
        Total time spent by all reduce tasks (ms)=228144
        Total vcore-seconds taken by all map tasks=89028
        Total vcore-seconds taken by all reduce tasks=228144
        Total megabyte-seconds taken by all map tasks=91164672
        Total megabyte-seconds taken by all reduce tasks=233619456
    Map-Reduce Framework
        Map input records=507535
        Map output records=4603237
        Map output bytes=42278460
        Map output materialized bytes=1240205
        Input split bytes=381
        Combine input records=4603237
        Combine output records=85622
        Reduce input groups=56726
        Reduce shuffle bytes=1240205
        Reduce input records=85622
        Reduce output records=139
        Spilled Records=171244
        Shuffled Maps =30
        Failed Shuffles=0
        Merged Map outputs=30
        GC time elapsed (ms)=2149
        CPU time spent (ms)=38070
        Physical memory (bytes) snapshot=2589450240
        Virtual memory (bytes) snapshot=20322811904
        Total committed heap usage (bytes)=1916796928
    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=26057865
    File Output Format Counters
        Bytes Written=611
```

4.3 StopWords with 10 reducers, combiner, and compression

Execution time 1’11’’. A slight improvement is observed but nothing major, maybe because the files are not big enough or the fact that it is only a pseudo distributed version : little to no time is gain in transfer, but time is lost in compressing and decompressing.

Job Overview	
Job Name:	StopWords
User Name:	cloudera
Queue:	root.cloudera
State:	SUCCEEDED
Uberized:	false
Submitted:	Thu Feb 16 14:56:00 PST 2017
Started:	Thu Feb 16 14:56:06 PST 2017
Finished:	Thu Feb 16 14:57:17 PST 2017
Elapsed:	1mins, 11sec
Diagnostics:	
Average Map Time	23sec
Average Shuffle Time	18sec
Average Merge Time	0sec
Average Reduce Time	3sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Thu Feb 16 14:56:01 PST 2017	quickstart.cloudera:8042	logs

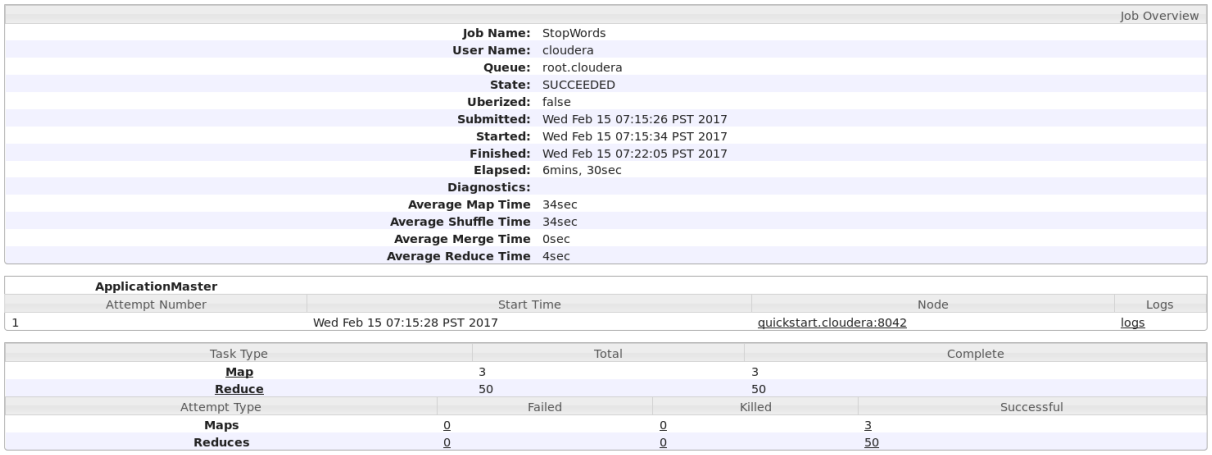
Task Type	Total	Complete
Map	3	3
Reduce	10	10

Attempt Type	Failed	Killed	Successful
Maps	0	0	3
Reduces	0	1	10

```
17/02/15 07:10:29 INFO mapreduce.Job: Job job_1485279029005_0019 completed
successfully
17/02/15 07:10:29 INFO mapreduce.Job: Counters: 50
    File System Counters
        FILE: Number of bytes read=1240085
        FILE: Number of bytes written=3999373
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=26058246
        HDFS: Number of bytes written=611
        HDFS: Number of read operations=39
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=20
    Job Counters
        Killed reduce tasks=2
        Launched map tasks=3
        Launched reduce tasks=12
        Data-local map tasks=3
        Total time spent by all maps in occupied slots (ms)=101483
        Total time spent by all reduces in occupied slots (ms)=419584
        Total time spent by all map tasks (ms)=101483
        Total time spent by all reduce tasks (ms)=419584
        Total vcore-seconds taken by all map tasks=101483
        Total vcore-seconds taken by all reduce tasks=419584
        Total megabyte-seconds taken by all map tasks=103918592
        Total megabyte-seconds taken by all reduce tasks=429654016
    Map-Reduce Framework
        Map input records=507535
        Map output records=4603237
        Map output bytes=42278460
        Map output materialized bytes=1240205
        Input split bytes=381
        Combine input records=4603237
        Combine output records=85622
        Reduce input groups=56726
        Reduce shuffle bytes=1240205
        Reduce input records=85622
        Reduce output records=139
        Spilled Records=171244
        Shuffled Maps =30
        Failed Shuffles=0
        Merged Map outputs=30
        GC time elapsed (ms)=2974
        CPU time spent (ms)=31130
        Physical memory (bytes) snapshot=1774018560
        Virtual memory (bytes) snapshot=19566510080
        Total committed heap usage (bytes)=1104359424
    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=26057865
    File Output Format Counters
        Bytes Written=611
```

4.4 StopWords with 50 reducers, combiner, and compression

Execution time 6’30’’ : adding this many reducers does not improve performance. The reason for this is that it is necessary for each Map task to create an intermediate file for each Reduce task, and if there are too many Reduce tasks the number of intermediate files explodes.



Job Overview

Job Name:	StopWords
User Name:	cloudera
Queue:	root.cloudera
State:	SUCCEEDED
Uberized:	false
Submitted:	Wed Feb 15 07:15:26 PST 2017
Started:	Wed Feb 15 07:15:34 PST 2017
Finished:	Wed Feb 15 07:22:05 PST 2017
Elapsed:	6mins, 30sec
Diagnostics:	
Average Map Time	34sec
Average Shuffle Time	34sec
Average Merge Time	0sec
Average Reduce Time	4sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Wed Feb 15 07:15:28 PST 2017	quickstart.cloudera:8042	logs

Task Type	Total	Complete
Map	3	3
Reduce	50	50

Attempt Type	Failed	Killed	Successful
Maps	0	0	3
Reduces	0	0	50

```
17/02/15 07:22:06 INFO mapreduce.Job: Job job_1485279029005_0020 completed
successfully
17/02/15 07:22:06 INFO mapreduce.Job: Counters: 50
    File System Counters
        FILE: Number of bytes read=1240325
        FILE: Number of bytes written=8674853
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=26058246
        HDFS: Number of bytes written=611
        HDFS: Number of read operations=159
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=100
    Job Counters
        Killed reduce tasks=2
        Launched map tasks=3
        Launched reduce tasks=50
        Data-local map tasks=3
        Total time spent by all maps in occupied slots (ms)=102506
        Total time spent by all reduces in occupied slots (ms)=1998484
        Total time spent by all map tasks (ms)=102506
        Total time spent by all reduce tasks (ms)=1998484
        Total vcore-seconds taken by all map tasks=102506
        Total vcore-seconds taken by all reduce tasks=1998484
        Total megabyte-seconds taken by all map tasks=104966144
        Total megabyte-seconds taken by all reduce tasks=2046447616
    Map-Reduce Framework
        Map input records=507535
        Map output records=4603237
        Map output bytes=42278460
        Map output materialized bytes=1240925
        Input split bytes=381
        Combine input records=4603237
        Combine output records=85622
        Reduce input groups=56726
        Reduce shuffle bytes=1240925
        Reduce input records=85622
        Reduce output records=139
        Spilled Records=171244
        Shuffled Maps =150
        Failed Shuffles=0
        Merged Map outputs=150
        GC time elapsed (ms)=11562
        CPU time spent (ms)=71060
        Physical memory (bytes) snapshot=6639992832
        Virtual memory (bytes) snapshot=79839502336
        Total committed heap usage (bytes)=3534434304
    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=26057865
    File Output Format Counters
        Bytes Written=611
```

4.5 INVERTED INDEX

Here are the performances of my inverted index. The counter `Reduce output records=55643` shows the number of unique words there are, excluding stopwords.

Job Overview	
Job Name:	InvertedIndex
User Name:	cloudera
Queue:	root.cloudera
State:	SUCCEEDED
Uberized:	false
Submitted:	Thu Feb 16 07:56:52 PST 2017
Started:	Thu Feb 16 07:57:06 PST 2017
Finished:	Thu Feb 16 08:09:56 PST 2017
Elapsed:	12mins, 49sec
Diagnostics:	
Average Map Time	31sec
Average Shuffle Time	18sec
Average Merge Time	0sec
Average Reduce Time	5mins, 56sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Thu Feb 16 07:57:00 PST 2017	quickstart.cloudera:8042	logs

Task Type	Total	Complete
Map	3	3
Reduce	10	10

Attempt Type	Failed	Killed	Successful
Maps	0	0	3
Reduces	0	1	10

```
17/02/16 08:09:58 INFO mapreduce.Job: Job job_1487258453206_0001 completed
successfully
17/02/16 08:09:59 INFO mapreduce.Job: Counters: 51
    File System Counters
        FILE: Number of bytes read=3761127
        FILE: Number of bytes written=7555863
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=60141048
        HDFS: Number of bytes written=1622956
        HDFS: Number of read operations=55821
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=20
    Job Counters
        Killed reduce tasks=1
        Launched map tasks=3
        Launched reduce tasks=11
        Data-local map tasks=3
        Total time spent by all maps in occupied slots (ms)=93941
        Total time spent by all reduces in occupied slots (ms)=4064026
        Total time spent by all map tasks (ms)=93941
        Total time spent by all reduce tasks (ms)=4064026
        Total vcore-seconds taken by all map tasks=93941
        Total vcore-seconds taken by all reduce tasks=4064026
        Total megabyte-seconds taken by all map tasks=96195584
        Total megabyte-seconds taken by all reduce tasks=4161562624
    Map-Reduce Framework
        Map input records=507535
        Map output records=4591847
        Map output bytes=74198099
        Map output materialized bytes=2313290
        Input split bytes=381
        Combine input records=4591847
        Combine output records=97518
        Reduce input groups=55782
        Reduce shuffle bytes=2313290
        Reduce input records=97518
        Reduce output records=55643
        Spilled Records=253971
        Shuffled Maps =30
        Failed Shuffles=0
        Merged Map outputs=30
        GC time elapsed (ms)=36668
        CPU time spent (ms)=1184860
        Physical memory (bytes) snapshot=3341504512
        Virtual memory (bytes) snapshot=20355305472
        Total committed heap usage (bytes)=2181562368
    Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
    invertedindex.InvertedIndex$CustomCounters
        UNIQUEWORDS=35726
    File Input Format Counters
        Bytes Read=26057865
    File Output Format Counters
        Bytes Written=1622956
```

5 CONCLUSION

This homework shows the influence of critical parameters of Hadoop implementation: number of reducers, combiners and compression, loading and saving files on hdfs, as well as the interesting features of counters.

SOURCES:

wiki.apache.org

Mining of Massive Datasets, Jure Leskovec, Anand Rajaraman, Jeff Ullman

Countless hours of web searches