



TU BERLIN

ADVANCED INFORMATION MANAGEMENT

HOMEWORK ASSIGNMENT 1

Programming in Hadoop and Clustering Exercises

Author:
Ward SCHODTS

Supervisor:
Juan SOTO

June 9, 2016

Programming in Hadoop

1. WordCount - "Hello World" of MapReduce

```
1 package de.tuberlin.dima.aim3.assignment1;
2
3 import de.tuberlin.dima.aim3.HadoopJob;
4 import org.apache.hadoop.fs.Path;
5 import org.apache.hadoop.io.IntWritable;
6 import org.apache.hadoop.io.Text;
7 import org.apache.hadoop.mapreduce.Job;
8 import org.apache.hadoop.mapreduce.Mapper;
9 import org.apache.hadoop.mapreduce.Reducer;
10 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
11 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
12
13 import java.io.IOException;
14 import java.util.ArrayList;
15 import java.util.Arrays;
16 import java.util.List;
17 import java.util.Map;
18 import java.util.function.Function;
19 import java.util.regex.Pattern;
20 import java.util.stream.StreamSupport;
21
22 import static java.util.stream.Collectors.counting;
23 import static java.util.stream.Collectors.groupingBy;
24
25 public class FilteringWordCount extends HadoopJob {
26
27     @Override
28     public int run(String[] args) throws Exception {
29         Map<String, String> parsedArgs = parseArgs(args);
30
31         Path inputPath = new Path(parsedArgs.get("--input"));
32         Path outputPath = new Path(parsedArgs.get("--output"));
33
34         Job wordCount = prepareJob(inputPath, outputPath, TextInputFormat.class,
35                                     FilteringWordCountMapper.class,
36                                     Text.class, IntWritable.class, WordCountReducer.class, Text.class,
37                                     IntWritable.class, TextOutputFormat.class);
38
39         wordCount.waitForCompletion(true);
40
41         return 0;
42     }
43
44     static class FilteringWordCountMapper extends Mapper<Object, Text, Text,
45                                     IntWritable> {
46
47         private ArrayList<String> filterList = new ArrayList<>();
48
49         /**
50          * Method to add words that should be filtered out.
51          *
52          * @param fl : the list with filtered words
53          */
54         public void addWordsToFilter(List<String> fl) {
55             this.filterList.addAll(fl);
56         }
57
58         public List getFilterList() {
59             return this.filterList;
60         }
61     }
62 }
```

```

60     @Override
61     protected void map(Object key, Text line, Context ctx) throws IOException,
InterruptedException {
62         String[] filterList = {"to", "and", "in", "the"};
63         addWordsToFilter(Arrays.asList(filterList));
64         Pattern.compile(" ").splitAsStream(line.toString().replace(",", ""))
        .map(String::toLowerCase).filter(1 -> !getFilterList().contains(1))
        .collect(groupingBy(Function.identity(), counting())).forEach((word, count) ->
        writeToCtx(word, count, ctx));
65     }
66
67     private void writeToCtx(String word, Long val, Context ctx) {
68         try {
69             ctx.write(new Text(word), new IntWritable(val.intValue()));
70         } catch (InterruptedException | IOException ignored) {
71
72         }
73     }
74
75 }
76
77 static class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable>
{
78
79     @Override
80     protected void reduce(Text key, Iterable<IntWritable> values, Context ctx)
81         throws IOException, InterruptedException {
82
83         ctx.write(key, new IntWritable(StreamSupport.stream(values.spliterator(),
84 false).mapToInt(IntWritable::get).sum()));
85     }
86
87 }
88 }

```

Listing 1: FilteringWordCount.java

```

1 16/06/08 23:46:45 WARN mapred.JobClient: Use GenericOptionsParser for parsing the
arguments. Applications should implement Tool for the same.
2 16/06/08 23:46:45 WARN mapred.JobClient: No job jar file set. User classes may not be
found. See JobConf(Class) or JobConf#setJar(String).
3 16/06/08 23:46:45 INFO input.FileInputFormat: Total input paths to process : 1
4 16/06/08 23:46:45 INFO mapred.JobClient: Running job: job_local_0001
5 16/06/08 23:46:45 INFO mapred.MapTask: io.sort.mb = 100
6 16/06/08 23:46:45 INFO mapred.MapTask: data buffer = 79691776/99614720
7 16/06/08 23:46:45 INFO mapred.MapTask: record buffer = 262144/327680
8 16/06/08 23:46:45 INFO mapred.MapTask: Starting flush of map output
9 16/06/08 23:46:45 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
10 16/06/08 23:46:45 INFO compress.CodecPool: Got brand-new compressor
11 16/06/08 23:46:45 INFO mapred.MapTask: Finished spill 0
12 16/06/08 23:46:45 INFO mapred.Task: Task:attempt_local_0001_m_000000_0 is done. And is
in the process of committing
13 16/06/08 23:46:46 INFO mapred.JobClient: map 0% reduce 0%
14 16/06/08 23:46:48 INFO mapred.LocalJobRunner:
15 16/06/08 23:46:48 INFO mapred.Task: Task 'attempt_local_0001_m_000000_0' done.
16 16/06/08 23:46:48 INFO mapred.LocalJobRunner:
17 16/06/08 23:46:48 INFO mapred.Merger: Merging 1 sorted segments
18 16/06/08 23:46:48 INFO compress.CodecPool: Got brand-new decompressor
19 16/06/08 23:46:48 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left
of total size: 80 bytes
20 16/06/08 23:46:48 INFO mapred.LocalJobRunner:
21 16/06/08 23:46:48 INFO mapred.Task: Task:attempt_local_0001_r_000000_0 is done. And is
in the process of committing
22 16/06/08 23:46:48 INFO mapred.LocalJobRunner:

```

```

23 16/06/08 23:46:48 INFO mapred.Task: Task attempt_local_0001_r_000000_0 is allowed to
    commit now
24 16/06/08 23:46:48 INFO output.FileOutputCommitter: Saved output of task '
    attempt_local_0001_r_000000_0' to /tmp/mahout-FilteringWordCountTest
    -1822392791331859456/output
25 16/06/08 23:46:49 INFO mapred.JobClient: map 100% reduce 0%
26 16/06/08 23:46:51 INFO mapred.LocalJobRunner: reduce > reduce
27 16/06/08 23:46:51 INFO mapred.Task: Task 'attempt_local_0001_r_000000_0' done.
28 16/06/08 23:46:52 INFO mapred.JobClient: map 100% reduce 100%
29 16/06/08 23:46:52 INFO mapred.JobClient: Job complete: job_local_0001
30 16/06/08 23:46:52 INFO mapred.JobClient: Counters: 16
31 16/06/08 23:46:52 INFO mapred.JobClient: Map-Reduce Framework
32 16/06/08 23:46:52 INFO mapred.JobClient: Combine output records=0
33 16/06/08 23:46:52 INFO mapred.JobClient: Spilled Records=34
34 16/06/08 23:46:52 INFO mapred.JobClient: Map output materialized bytes=84
35 16/06/08 23:46:52 INFO mapred.JobClient: Reduce input records=17
36 16/06/08 23:46:52 INFO mapred.JobClient: Reduce output records=9
37 16/06/08 23:46:52 INFO mapred.JobClient: Map input records=4
38 16/06/08 23:46:52 INFO mapred.JobClient: SPLIT_RAW_BYTES=133
39 16/06/08 23:46:52 INFO mapred.JobClient: Map output records=17
40 16/06/08 23:46:52 INFO mapred.JobClient: Map output bytes=153
41 16/06/08 23:46:52 INFO mapred.JobClient: Reduce shuffle bytes=0
42 16/06/08 23:46:52 INFO mapred.JobClient: Combine input records=0
43 16/06/08 23:46:52 INFO mapred.JobClient: Reduce input groups=9
44 16/06/08 23:46:52 INFO mapred.JobClient: File Input Format Counters
45 16/06/08 23:46:52 INFO mapred.JobClient: Bytes Read=107
46 16/06/08 23:46:52 INFO mapred.JobClient: FileSystemCounters
47 16/06/08 23:46:52 INFO mapred.JobClient: FILE.BYTES-WRITTEN=64694
48 16/06/08 23:46:52 INFO mapred.JobClient: FILE.BYTES-READ=672
49 16/06/08 23:46:52 INFO mapred.JobClient: File Output Format Counters
50 16/06/08 23:46:52 INFO mapred.JobClient: Bytes Written=78
51
52 Process finished with exit code 0

```

Listing 2: Output FilteringWordCount

2. A custom Writable

```

1 package de.tuberlin.dima.aim3.assignment1;
2
3 import org.apache.hadoop.hdfs.util.ByteArray;
4 import org.apache.hadoop.io.Writable;
5
6 import java.io.DataInput;
7 import java.io.DataOutput;
8 import java.io.IOException;
9 import java.util.Arrays;
10
11 public class PrimeNumbersWritable implements Writable {
12
13     private int[] numbers;
14
15     public PrimeNumbersWritable() {
16         numbers = new int[0];
17     }
18
19     public PrimeNumbersWritable(int... numbers) {
20         this.numbers = numbers;
21     }
22
23     @Override
24     public void write(DataOutput out) throws IOException {
25         out.writeInt(numbers.length);
26         Arrays.stream(numbers).forEach((v) -> writeToOut(out, v));
27     }
28
29     private void writeToOut(DataOutput out, int i) {

```

```

30         try {
31             out.writeInt(i);
32         } catch (IOException ignored) {
33         }
34     }
35
36
37     @Override
38     public void readFields(DataInput in) throws IOException {
39         int length = in.readInt();
40
41         int[] temp = new int[length];
42         for (int i = 0; i < length; i++) {
43             temp[i] = in.readInt();
44         }
45         this.numbers = temp;
46     }
47
48     @Override
49     public boolean equals(Object obj) {
50         if (obj instanceof PrimeNumbersWritable) {
51             PrimeNumbersWritable other = (PrimeNumbersWritable) obj;
52             return Arrays.equals(numbers, other.numbers);
53         }
54         return false;
55     }
56
57     @Override
58     public int hashCode() {
59         return Arrays.hashCode(numbers);
60     }
61 }

```

Listing 3: PrimeNumbersWritable.java

```

1 Process finished with exit code 0

```

Listing 4: Output PrimeNumbersWritable

3. Average temperature per month

```

1 package de.tuberlin.dima.aim3.assignment1;
2
3
4 import de.tuberlin.dima.aim3.HadoopJob;
5 import org.apache.hadoop.conf.Configuration;
6 import org.apache.hadoop.fs.Path;
7 import org.apache.hadoop.io.DoubleWritable;
8 import org.apache.hadoop.io.IntWritable;
9 import org.apache.hadoop.io.Text;
10 import org.apache.hadoop.mapreduce.Mapper;
11 import org.apache.hadoop.mapreduce.Reducer;
12
13 import java.io.IOException;
14 import java.util.Map;
15
16 import org.apache.hadoop.mapreduce.Job;
17 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
18 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
19
20 public class AverageTemperaturePerMonth extends HadoopJob {
21
22     @Override
23     public int run(String[] args) throws Exception {
24         Map<String, String> parsedArgs = parseArgs(args);
25
26         Path inputPath = new Path(parsedArgs.get("--input"));

```

```

27     Path outputPath = new Path(parsedArgs.get("--output"));
28
29     double minimumQuality = Double.parseDouble(parsedArgs.get("--minimumQuality"));
30
31     Job averageTemperature = prepareJob(inputPath, outputPath, TextInputFormat.
class ,
32         AverageTemperatureMapper.class, Text.class, IntWritable.class,
33         AverageTemperatureReducer.class, Text.class, DoubleWritable.class,
34         TextOutputFormat.class);
35
36     averageTemperature.getConfiguration().set("minimumQuality", String.valueOf(
minimumQuality));
37
38     averageTemperature.waitForCompletion(true);
39
40     return 0;
41 }
42
43
44 static class AverageTemperatureMapper extends Mapper<Object, Text, Text,
IntWritable> {
45
46
47     @Override
48     protected void map(Object key, Text line, Context ctx) throws IOException,
InterruptedException {
49         Configuration conf = ctx.getConfiguration();
50         double minimumQuality = Double.parseDouble(conf.get("minimumQuality"));
51
52         String l = line.toString();
53         String[] ls = l.split("\t");
54         if (minimumQuality <= Double.parseDouble(ls[ls.length - 1])) {
55             String K = ls[0] + "\t" + ls[1];
56             ctx.write(new Text(K), new IntWritable(Integer.parseInt(ls[2])));
57         }
58     }
59 }
60
61 }
62
63 static class AverageTemperatureReducer extends Reducer<Text, IntWritable, Text,
DoubleWritable> {
64     @Override
65     protected void reduce(Text key, Iterable<IntWritable> values, Context ctx)
throws IOException, InterruptedException {
66         int sum = 0;
67         int length = 0;
68         for (IntWritable value : values) {
69             sum += value.get();
70             length++;
71         }
72         double average = (double) sum / length;
73         ctx.write(new Text(key), new DoubleWritable(average));
74     }
75 }
76 }
77 }

```

Listing 5: AverageTemperaturePerMonth.java

```

1 16/06/08 23:57:44 WARN mapred.JobClient: Use GenericOptionsParser for parsing the
arguments. Applications should implement Tool for the same.
2 16/06/08 23:57:44 WARN mapred.JobClient: No job jar file set. User classes may not be
found. See JobConf(Class) or JobConf#setJar(String).
3 16/06/08 23:57:44 INFO input.FileInputFormat: Total input paths to process : 1
4 16/06/08 23:57:44 INFO mapred.JobClient: Running job: job_local-0001
5 16/06/08 23:57:44 INFO mapred.MapTask: io.sort.mb = 100

```

```

6 16/06/08 23:57:44 INFO mapred.MapTask: data buffer = 79691776/99614720
7 16/06/08 23:57:44 INFO mapred.MapTask: record buffer = 262144/327680
8 16/06/08 23:57:44 INFO mapred.MapTask: Starting flush of map output
9 16/06/08 23:57:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
  your platform... using builtin-java classes where applicable
10 16/06/08 23:57:44 INFO compress.CodecPool: Got brand-new compressor
11 16/06/08 23:57:44 INFO mapred.MapTask: Finished spill 0
12 16/06/08 23:57:44 INFO mapred.Task: Task:attempt_local_0001_m_000000_0 is done. And is
  in the process of committing
13 16/06/08 23:57:45 INFO mapred.JobClient: map 0% reduce 0%
14 16/06/08 23:57:47 INFO mapred.LocalJobRunner:
15 16/06/08 23:57:47 INFO mapred.Task: Task 'attempt_local_0001_m_000000_0' done.
16 16/06/08 23:57:47 INFO mapred.LocalJobRunner:
17 16/06/08 23:57:47 INFO mapred.Merger: Merging 1 sorted segments
18 16/06/08 23:57:47 INFO compress.CodecPool: Got brand-new decompressor
19 16/06/08 23:57:47 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left
  of total size: 14587 bytes
20 16/06/08 23:57:47 INFO mapred.LocalJobRunner:
21 16/06/08 23:57:47 INFO mapred.Task: Task:attempt_local_0001_r_000000_0 is done. And is
  in the process of committing
22 16/06/08 23:57:47 INFO mapred.LocalJobRunner:
23 16/06/08 23:57:47 INFO mapred.Task: Task attempt_local_0001_r_000000_0 is allowed to
  commit now
24 16/06/08 23:57:47 INFO output.FileOutputCommitter: Saved output of task '
  attempt_local_0001_r_000000_0' to /tmp/mahout-AverageTemperaturePerMonthTest
  -2969850880617501696/output
25 16/06/08 23:57:48 INFO mapred.JobClient: map 100% reduce 0%
26 16/06/08 23:57:50 INFO mapred.LocalJobRunner: reduce > reduce
27 16/06/08 23:57:50 INFO mapred.Task: Task 'attempt_local_0001_r_000000_0' done.
28 16/06/08 23:57:51 INFO mapred.JobClient: map 100% reduce 100%
29 16/06/08 23:57:51 INFO mapred.JobClient: Job complete: job_local_0001
30 16/06/08 23:57:51 INFO mapred.JobClient: Counters: 16
31 16/06/08 23:57:51 INFO mapred.JobClient: Map-Reduce Framework
32 16/06/08 23:57:51 INFO mapred.JobClient: Combine output records=0
33 16/06/08 23:57:51 INFO mapred.JobClient: Spilled Records=14988
34 16/06/08 23:57:51 INFO mapred.JobClient: Map output materialized bytes=14591
35 16/06/08 23:57:51 INFO mapred.JobClient: Reduce input records=7494
36 16/06/08 23:57:51 INFO mapred.JobClient: Reduce output records=72
37 16/06/08 23:57:51 INFO mapred.JobClient: Map input records=10000
38 16/06/08 23:57:51 INFO mapred.JobClient: SPLIT_RAW_BYTES=149
39 16/06/08 23:57:51 INFO mapred.JobClient: Map output records=7494
40 16/06/08 23:57:51 INFO mapred.JobClient: Map output bytes=84301
41 16/06/08 23:57:51 INFO mapred.JobClient: Reduce shuffle bytes=0
42 16/06/08 23:57:51 INFO mapred.JobClient: Combine input records=0
43 16/06/08 23:57:51 INFO mapred.JobClient: Reduce input groups=72
44 16/06/08 23:57:51 INFO mapred.JobClient: File Input Format Counters
45 16/06/08 23:57:51 INFO mapred.JobClient: Bytes Read=192657
46 16/06/08 23:57:51 INFO mapred.JobClient: FileSystemCounters
47 16/06/08 23:57:51 INFO mapred.JobClient: FILE_BYTES_WRITTEN=95989
48 16/06/08 23:57:51 INFO mapred.JobClient: FILE_BYTES_READ=400317
49 16/06/08 23:57:51 INFO mapred.JobClient: File Output Format Counters
50 16/06/08 23:57:51 INFO mapred.JobClient: Bytes Written=1769
51
52 Process finished with exit code 0

```

Listing 6: Output AverageTemperature

Clustering

1. Metrics I

We have 3 points A(4,8), B(9,5) and C(2,2).

The *centroid* is then:

$$(x, y) = \left(\frac{4 + 9 + 2}{3}, \frac{8 + 5 + 2}{3} \right) = (5, 5)$$

We now calculate the error towards the centroid for every point:

A(4,8):

$$\sqrt{(4 - 5)^2 + (8 - 5)^2} = \sqrt{10}$$

B(9,5):

$$\sqrt{(9 - 5)^2 + (5 - 5)^2} = \sqrt{16} = 4$$

C(2,2):

$$\sqrt{(2 - 5)^2 + (2 - 5)^2} = \sqrt{18} = 3 \cdot \sqrt{2}$$

The SSE is then:

$$\sqrt{10}^2 + 4^2 + 3 \cdot \sqrt{2}^2 = 44$$

2. Metrics II

If you partition 3 points into 2 clusters, there's one with one element and one with two. The SSE of 1 point is always 0. So the total SSE is only determined by the SSE of the possible combinations of two points.

$$Centroid((3, 0), (0, 7)) = (1.5, 3.5)$$

$$SSE((3, 0), (0, 7)) = \sqrt{2.25 + 12.25}^2 + \sqrt{2.25 + 12.25}^2 = 29$$

$$Centroid(((0, 7), (6, 5))) = (3, 6)$$

$$SSE(((0, 7), (6, 5))) = \sqrt{9 + 1}^2 + \sqrt{9 + 1}^2 = 20$$

$$Centroid((3, 0), (6, 5)) = (4.5, 2.5)$$

$$SSE((3, 0), (6, 5)) = \sqrt{2.25 + 6.25}^2 + \sqrt{2.25 + 6.25}^2 = 17$$

The SSE is the smallest in the last configuration therefore the the most optimal split in two groups is the following:

Cluster 1: [(0,7)]

Cluster 2: [(3,0),(6,5)]