

Big Data Infrastructures

Fall 2018

Lab 05 : Hadoop and MapReduce

Author : Thomas Schaller, Sylvain Julmy

Professor : Philippe Cudré-Mauroux

Running the experiment

In order to easily run the experiments, we develop a small bash script (see ??) which compile, run, clean and move the output of the experiments to our local directory in order to watch them.

Wordcount :

Listing 2, 3 and 4 shows the code for the various version of the Wordcount. The function to initialize the map used for *htmlEntities* is shown in listing 5.

Quadruples :

Listing 6 shows the code for counting the number of literals for each node, and listing 7 shows the code for computing the In and Out degree of each node. Note that we have also created a class *IntTriple* (in listing 10) to simplify and pass Triple of int between the map and the reduce parts.

Outputs :

The various outputs of our programm are available under the following HDFS folder :

`/bdi_2018/bdi18_07/ex${exercice_number}_output`

```

1  #!/bin/bash
2
3  # usage :
4  # 1st argument is the name of the run
5  # 2nd argument is the path of the java file
6  # 3rd argument is the path for the input files
7  # example ./run.sh ex1 WordCountEx1.java /bdi_2018/data/NYTimes_articles
8
9  experiment=$1
10 javaFile=$2
11 inPath=$3
12
13 filename=$(basename -- "$javaFile")
14 filename="${filename%.*}"
15 classFile="${filename}.class"
16 jarFile="${filename}.jar"
17
18 # for telegram
19 apiToken="651009095:AAHTl2_1LEyJNAVsuXtdhCzF5jIiGnDAsOY"
20 chatId="23817760"
21
22 # export java...
23 export JAVA_HOME=~/.bundle/java_7
24 export PATH=$PATH:${JAVA_HOME}/bin
25
26 # run experiment
27 javac -classpath `bin/hadoop classpath` ${jarFile}
28 jar cf ${jarFile} ${filename}*.class
29 bin/hadoop jar ${jarFile} ${filename} ${inPath} /bdi_2018/bdi18_07/${experiment}_output
30 bin/hadoop fs -copyToLocal /bdi_2018/bdi18_07/${experiment}_output ~/${experiment}_output
31
32 # uncomment to remove the output from HDFS
33 # bin/hadoop fs -rm -r -skipTrash /bdi_2018/bdi18_07/output
34
35 send() {
36     curl -s \
37     -X POST \
38     https://api.telegram.org/bot${apiToken}/sendMessage \
39     -d text="$1" \
40     -d chat_id=${chatId}
41 }
42
43 rm *.jar
44 rm *.class
45
46 send "experiment $experiment done"

```

Listing 1: Script to run the exercises on the Hadoop cluster.

```

1  public class WordCountEx1 {
2
3      public static void main(String[] args) throws Exception {
4          Configuration conf = new Configuration();
5          Job job = Job.getInstance(conf, "word count");
6          job.setJar("WordCountEx1.jar");
7          job.setJarByClass(WordCountEx1.class);
8          job.setMapperClass(TokenizerMapper.class);
9          job.setCombinerClass(IntSumReducer.class);
10         job.setReducerClass(IntSumReducer.class);
11         job.setOutputKeyClass(Text.class);
12         job.setOutputValueClass(IntWritable.class);
13         FileInputFormat.addInputPath(job, new Path(args[0]));
14         FileOutputFormat.setOutputPath(job, new Path(args[1]));
15         System.exit(job.waitForCompletion(true) ? 0 : 1);
16     }
17
18     public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
19         private final static IntWritable one = new IntWritable(1);
20         private Text word = new Text();
21
22         public void map(
23             Object key,
24             Text value,
25             Context context
26         ) throws IOException, InterruptedException {
27
28             StringTokenizer itr = new StringTokenizer(value.toString());
29
30             while (itr.hasMoreTokens()) {
31                 word.set(itr.nextToken());
32                 context.write(word, one);
33             }
34         }
35     }
36
37     public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
38         private IntWritable result = new IntWritable();
39
40         public void reduce(
41             Text key,
42             Iterable<IntWritable> values,
43             Context context
44         ) throws IOException, InterruptedException {
45
46             int sum = 0;
47             for (IntWritable val : values) {
48                 sum += val.get();
49             }
50             result.set(sum);
51             context.write(key, result);
52         }
53     }
54 }

```

Listing 2: First implementation for Wordcount.

```

1 public class WordCountEx2 {
2
3     public static void main(String[] args) throws Exception {
4         Configuration conf = new Configuration();
5         Job job = Job.getInstance(conf, "word count");
6         job.setJar("WordCountEx2.jar");
7         job.setJarByClass(WordCountEx2.class);
8         job.setMapperClass(TokenizerMapper.class);
9         job.setCombinerClass(IntSumReducer.class);
10        job.setReducerClass(IntSumReducer.class);
11        job.setOutputKeyClass(Text.class);
12        job.setOutputValueClass(IntWritable.class);
13        FileInputFormat.addInputPath(job, new Path(args[0]));
14        FileOutputFormat.setOutputPath(job, new Path(args[1]));
15        System.exit(job.waitForCompletion(true) ? 0 : 1);
16    }
17
18    public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
19        private final static IntWritable one = new IntWritable(1);
20        private Text word = new Text();
21
22        public void map(
23            Object key,
24            Text value,
25            Context context
26        ) throws IOException, InterruptedException {
27
28            String value2 = value.toString();
29            value2 = value2.replaceAll("[,\\.:\\]\\\"@?!]", "")
30                .replaceAll("&rdquo;", "")
31                .toLowerCase();
32
33            StringTokenizer itr = new StringTokenizer(value2);
34            while (itr.hasMoreTokens()) {
35                word.set(itr.nextToken());
36                context.write(word, one);
37            }
38        }
39    }
40
41    public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
42        private IntWritable result = new IntWritable();
43
44        public void reduce(
45            Text key,
46            Iterable<IntWritable> values,
47            Context context
48        ) throws IOException, InterruptedException {
49
50            int sum = 0;
51            for (IntWritable val : values) {
52                sum += val.get();
53            }
54            result.set(sum);
55            context.write(key, result);
56        }
57    }
58 }

```

Listing 3: First improvement for Wordcount.

```

1 public class WordCountEx3 {
2
3     public static void main(String[] args) throws Exception {
4         Configuration conf = new Configuration();
5         Job job = Job.getInstance(conf, "word count");
6         job.setJar("WordCountEx3.jar");
7         job.setJarByClass(WordCountEx3.class);
8         job.setMapperClass(TokenizerMapper.class);
9         job.setCombinerClass(IntSumReducer.class);
10        job.setReducerClass(IntSumReducer.class);
11        job.setOutputKeyClass(Text.class);
12        job.setOutputValueClass(IntWritable.class);
13        FileInputFormat.addInputPath(job, new Path(args[0]));
14        FileOutputFormat.setOutputPath(job, new Path(args[1]));
15        System.exit(job.waitForCompletion(true) ? 0 : 1);
16    }
17
18    public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
19        private final static IntWritable one = new IntWritable(1);
20        public HashMap<String, String> htmlEntities;
21        private Text word = new Text();
22
23        public void map(
24            Object key,
25            Text value,
26            Context context
27        ) throws IOException, InterruptedException {
28
29            initHashMap();
30            String value2 = value.toString();
31            value2 = value2.replaceAll("[,\\.:()\\\"@?!]", "");
32            value2 = value2.replaceAll("&rdquo;", "");
33            value2 = value2.toLowerCase();
34            for (Map.Entry<String, String> entry : htmlEntities.entrySet()) {
35                String key1 = entry.getKey();
36                String value1 = entry.getValue();
37                value2 = value2.replaceAll(key1, value1);
38            }
39            StringTokenizer itr = new StringTokenizer(value2);
40            while (itr.hasMoreTokens()) {
41                word.set(itr.nextToken());
42                context.write(word, one);
43            }
44        }
45    }
46
47    public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
48        private IntWritable result = new IntWritable();
49
50        public void reduce(
51            Text key,
52            Iterable<IntWritable> values,
53            Context context
54        ) throws IOException, InterruptedException {
55
56            int sum = 0;
57            for (IntWritable val : values) {
58                sum += val.get();
59            }
60            result.set(sum);
61            context.write(key, result);
62        }
63    }
64 }

```

Listing 4: First improvement for Wordcount.

```

1  public void initHashMap() {
2      htmlEntities = new HashMap<String, String>();
3      htmlEntities.put("<", "<"); htmlEntities.put(">", ">");
4      htmlEntities.put("&", "&"); htmlEntities.put(""", "\"");
5      htmlEntities.put("à", "à"); htmlEntities.put("À", "À");
6      htmlEntities.put("â", "â"); htmlEntities.put("ä", "ä");
7      htmlEntities.put("Ä", "Ä"); htmlEntities.put("Â", "Â");
8      htmlEntities.put("å", "å"); htmlEntities.put("Å", "Å");
9      htmlEntities.put("æ", "æ"); htmlEntities.put("Æ", "Æ");
10     htmlEntities.put("ç", "ç"); htmlEntities.put("Ç", "Ç");
11     htmlEntities.put("é", "é"); htmlEntities.put("É", "É");
12     htmlEntities.put("è", "è"); htmlEntities.put("È", "È");
13     htmlEntities.put("ê", "ê"); htmlEntities.put("Ê", "Ê");
14     htmlEntities.put("ë", "ë"); htmlEntities.put("Ë", "Ë");
15     htmlEntities.put("ï", "ï"); htmlEntities.put("Ï", "Ï");
16     htmlEntities.put("ô", "ô"); htmlEntities.put("Ô", "Ô");
17     htmlEntities.put("ö", "ö"); htmlEntities.put("Ö", "Ö");
18     htmlEntities.put("ø", "ø"); htmlEntities.put("Ø", "Ø");
19     htmlEntities.put("ß", "ß"); htmlEntities.put("ù", "ù");
20     htmlEntities.put("Ù", "Ù"); htmlEntities.put("û", "û");
21     htmlEntities.put("Û", "Û"); htmlEntities.put("ü", "ü");
22     htmlEntities.put("Ü", "Ü"); htmlEntities.put(" ", " ");
23     htmlEntities.put("©", "©"); htmlEntities.put("®", "®");
24     htmlEntities.put("€", "€"); htmlEntities.put("’", "’");
25 }

```

Listing 5: Initialization of *htmlEntities*.

```

1 public class QuadruplesCounter {
2
3     public static void main(String[] args) throws Exception {
4         Configuration conf = new Configuration();
5         Job job = Job.getInstance(conf, "quadruples count");
6         job.setJar("QuadruplesCounter.jar");
7         job.setJarByClass(QuadruplesCounter.class);
8         job.setMapperClass(TokenizerMapper.class);
9         job.setCombinerClass(IntSumReducer.class);
10        job.setReducerClass(IntSumReducer.class);
11        job.setOutputKeyClass(Text.class);
12        job.setOutputValueClass(IntWritable.class);
13        FileInputFormat.addInputPath(job, new Path(args[0]));
14        FileOutputFormat.setOutputPath(job, new Path(args[1]));
15        System.exit(job.waitForCompletion(true) ? 0 : 1);
16    }
17
18    // Test if s is a valid URI or not
19    private static boolean isValidURI(String s) {
20        try {
21            new URI(s);
22            return true;
23        } catch (URISyntaxException e) {
24            return false;
25        }
26    }
27
28    public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
29
30        private final static IntWritable one = new IntWritable(1);
31        private Text word = new Text();
32
33        public void map(Object key, Text value, Context context)
34            throws IOException, InterruptedException {
35
36            String[] values = value.toString().split("\\t");
37            String subject = values[0];
38            String predicate = values[1];
39            String object = values[2];
40            String provenance = values[3];
41
42            assert isValidURI(subject);
43            assert isValidURI(predicate);
44            assert isValidURI(provenance);
45
46            if (!isValidURI(object)) { // object is a literal
47                word.set(subject);
48                context.write(word, one);
49            }
50        }
51    }
52
53    public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
54        private IntWritable result = new IntWritable();
55
56        public void reduce(Text key, Iterable<IntWritable> values, Context context)
57            throws IOException, InterruptedException {
58            int sum = 0;
59            for (IntWritable val : values) {
60                sum += val.get();
61            }
62
63            if (sum >= 5) {
64                result.set(sum);
65                context.write(key, result);
66            }
67        }
68    }
69 }

```

Listing 6: Map-Reduce program for counting the number of literals for each node.

```

1 public class InOutDegreeCounter {
2
3     public static void main(String[] args) throws Exception {
4         Configuration conf = new Configuration();
5
6         Job job = Job.getInstance(conf, "in and out degree count");
7
8         job.setJar("InOutDegreeCounter.jar");
9         job.setJarByClass(InOutDegreeCounter.class);
10
11         job.setMapperClass(TokenizerMapper.class);
12         job.setCombinerClass(IntSumReducer.class);
13         job.setReducerClass(IntSumReducer.class);
14
15         // Map output
16         job.setMapOutputKeyClass(Text.class);
17         job.setMapOutputValueClass(IntTriple.class);
18
19         // Global output
20         job.setOutputKeyClass(Text.class);
21         job.setOutputValueClass(IntTriple.class);
22
23         FileInputFormat.addInputPath(job, new Path(args[0]));
24         FileOutputFormat.setOutputPath(job, new Path(args[1]));
25         System.exit(job.waitForCompletion(true) ? 0 : 1);
26     }
27
28     // Test if s is a valid URI or not
29     private static boolean isValidURI(String s) {
30         try {
31             URI uri = new URI(s);
32             return true;
33         } catch (URISyntaxException e) {
34             return false;
35         }
36     }
37 }

```

Listing 7: Configuration for the last exercise and a small method to check URI.


```

1 public static class TokenizerMapper extends Mapper<Object, Text, Text, IntTriple> {
2
3     private Text word = new Text();
4
5     public void map(Object key, Text value, Context context)
6         throws IOException, InterruptedException {
7
8         String[] values = value.toString().split("\\t");
9         String subject = values[0];
10        String predicate = values[1];
11        String object = values[2];
12        String provenance = values[3];
13
14        assert isValidURI(subject);
15        assert isValidURI(predicate);
16        assert isValidURI(provenance);
17
18        // Structure for the triple is (#Literal,#In,#Out)
19
20        if (isValidURI(object)) { // object is a valid URI
21            word.set(subject);
22            // increment the out count
23            IntTriple triple = new IntTriple(0, 0, 1);
24            context.write(word, triple);
25        } else { // object is a literal
26            word.set(subject);
27            // increment the literal count and the out count
28            IntTriple triple = new IntTriple(1, 0, 1);
29            context.write(word, triple);
30        }
31
32        word.set(object);
33        // increment the in count for the object
34        IntTriple triple2 = new IntTriple(0, 1, 0);
35        context.write(word, triple2);
36    }
37 }

```

Listing 8: Mapper class for the last exercise.

```

1 public static class IntSumReducer extends Reducer<Text, IntTriple, Text, IntTriple> {
2     public void reduce(Text key, Iterable<IntTriple> values, Context context)
3         throws IOException, InterruptedException {
4
5         int sumLiteral = 0, sumIn = 0, sumOut = 0;
6
7         for (IntTriple triple : values) {
8             sumLiteral += triple.a.get();
9             sumIn += triple.b.get();
10            sumOut += triple.c.get();
11        }
12
13        assert isValidURI(key.toString());
14
15        if (sumLiteral >= 10) {
16            context.write(key, new IntTriple(sumLiteral, sumIn, sumOut));
17        }
18    }
19 }

```

Listing 9: Reducer class for the last exercise.

```

1  static class IntTriple implements Writable {
2
3      private IntWritable a;
4      private IntWritable b;
5      private IntWritable c;
6
7      public IntTriple() {
8          set(new IntWritable(0), new IntWritable(0), new IntWritable(0));
9      }
10
11     public IntTriple(IntWritable a, IntWritable b, IntWritable c) {
12         this.a = a;
13         this.b = b;
14         this.c = c;
15     }
16
17     public IntTriple(int a, int b, int c) {
18         this.a = new IntWritable(a);
19         this.b = new IntWritable(b);
20         this.c = new IntWritable(c);
21     }
22
23     public int compareTo(IntTriple that) {
24         int cmp = this.a.compareTo(that.a);
25         if (cmp != 0) return cmp;
26         cmp = this.b.compareTo(that.b);
27         if (cmp != 0) return cmp;
28         return this.c.compareTo(that.c);
29     }
30
31     public void set(IntWritable a, IntWritable b, IntWritable c) {
32         this.a = a;
33         this.b = b;
34         this.c = c;
35     }
36
37     public void write(DataOutput dataOutput) throws IOException {
38         a.write(dataOutput);
39         b.write(dataOutput);
40         c.write(dataOutput);
41     }
42
43     public void readFields(DataInput dataInput) throws IOException {
44         a.readFields(dataInput);
45         b.readFields(dataInput);
46         c.readFields(dataInput);
47     }
48
49     @Override
50     public int hashCode() {
51         return a.hashCode() * 163 * 163 + b.hashCode() * 163 + c.hashCode();
52     }
53
54     @Override
55     public boolean equals(Object obj) {
56         if (obj instanceof IntTriple) {
57             IntTriple that = (IntTriple) obj;
58             return this.a.equals(that.a) &&
59                    this.b.equals(that.b) &&
60                    this.c.equals(that.c);
61         }
62         return false;
63     }
64
65     @Override
66     public String toString() {
67         return String.format("(%d,%d,%d)", a.get(), b.get(), c.get());
68     }
69 }

```

Listing 10: Implementation of a IntTriple class which represent a Triple of IntWritable.