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 easely 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${exerice_number}_output`

```bash
#! /bin/bash

# usage :
# 1st argument is the name of the run
# 2nd argument is the path of the java file
# 3nd argument is the path for the input files
# example ./run.sh ex1 WordCountEx1.java /bdi_2018/data/NYTimes_articles

experiment=$1
javaFile=$2
inPath=$3

filename=$(basename -- "$javaFile")
filename="${filename%.*}"
classFile="$filename.class"
jarFile="$filename.jar"

# for telegram
apiToken="651009095:AAHTl2_1LEyJNAVsuxtdhCzF5jIiGnDAs0Y"
chatId="23817760"

# export java...
export JAVA_HOME=~/bundle/java_7
export PATH=$PATH:${JAVA_HOME}/bin

# run experiment
javac -classpath `bin/hadoop classpath` ${javaFile}
jar cf ${jarFile} ${filename}*class
bin/hadoop jar ${jarFile} ${filename} ${inPath} /bdi_2018/bdi18_07/${experiment}_output
bin/hadoop fs -copyToLocal /bdi_2018/bdi18_07/${experiment}_output ~/${experiment}_output

# uncomment to remove the output from HDFS
# bin/hadoop fs -rm -r -skipTrash /bdi_2018/bdi18_07/output

send() {
    curl -s \
    -X POST \
    https://api.telegram.org/bot${apiToken}/sendMessage \
    -d text="$1" \
    -d chat_id=${chatId}
}

rm *.jar
rm *.class

send "experiment $experiment done"
```

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

```java
public class WordCountEx1 {

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJar("WordCountEx1.jar");
        job.setJarByClass(WordCountEx1.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }

    public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

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

            StringTokenizer itr = new StringTokenizer(value.toString());

            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

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

            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }
}
```

Listing 2: First implementation for Wordcount.

```java
public class WordCountEx2 {

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJar("WordCountEx2.jar");
        job.setJarByClass(WordCountEx2.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }

    public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

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

            String value2 = value.toString();
            value2 = value2.replaceAll("[,\\.:()\"@?!]", "")
                           .replaceAll("&rdquo;", "")
                           .toLowerCase();

            StringTokenizer itr = new StringTokenizer(value2);
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

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

            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }
}
```

Listing 3: First improvement for Wordcount.

```java
public class WordCountEx3 {

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJar("WordCountEx3.jar");
        job.setJarByClass(WordCountEx3.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }

    public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        public HashMap<String, String> htmlEntities;
        private Text word = new Text();

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

            initHashMap();
            String value2 = value.toString();
            value2 = value2.replaceAll("[,\\.:()\"@?!]", "")
                           .replaceAll("&rdquo;", "")
                           .toLowerCase();
            for (Map.Entry<String, String> entry : htmlEntities.entrySet()) {
                String key1 = entry.getKey();
                String value1 = entry.getValue();
                value2 = value2.replaceAll(key1, value1);
            }
            StringTokenizer itr = new StringTokenizer(value2);
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

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

            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }
}
```


Listing 4: First improvement for Wordcount.

```
1   public void initHashMap() {
2           htmlEntities = new HashMap<String, String>();
3           htmlEntities.put("&lt;", "<"); htmlEntities.put("&gt;", ">");
4           htmlEntities.put("&amp;", "&"); htmlEntities.put("&quot;", "\"");
5           htmlEntities.put("&agrave;", "à"); htmlEntities.put("&Agrave;", "À");
6           htmlEntities.put("&acirc;", "â"); htmlEntities.put("&auml;", "ä");
7           htmlEntities.put("&Auml;", "Ä"); htmlEntities.put("&Acirc;", "Â");
8           htmlEntities.put("&aring;", "å"); htmlEntities.put("&Aring;", "Å");
9           htmlEntities.put("&aelig;", "æ"); htmlEntities.put("&AElig;", "Æ");
10          htmlEntities.put("&ccedil;", "ç"); htmlEntities.put("&Ccedil;", "Ç");
11          htmlEntities.put("&eacute;", "é"); htmlEntities.put("&Eacute;", "É");
12          htmlEntities.put("&egrave;", "è"); htmlEntities.put("&Egrave;", "È");
13          htmlEntities.put("&ecirc;", "ê"); htmlEntities.put("&Ecirc;", "Ê");
14          htmlEntities.put("&euml;", "ë"); htmlEntities.put("&Euml;", "Ë");
15          htmlEntities.put("&iuml;", "ï"); htmlEntities.put("&Iuml;", "Ï");
16          htmlEntities.put("&ocirc;", "ô"); htmlEntities.put("&Ocirc;", "Ô");
17          htmlEntities.put("&ouml;", "ö"); htmlEntities.put("&Ouml;", "Ö");
18          htmlEntities.put("&oslash;", "ø"); htmlEntities.put("&Oslash;", "Ø");
19          htmlEntities.put("&szlig;", "ß"); htmlEntities.put("&ugrave;", "ù");
20          htmlEntities.put("&Ugrave;", "Ù"); htmlEntities.put("&ucirc;", "û");
21          htmlEntities.put("&Ucirc;", "Û"); htmlEntities.put("&uuml;", "ü");
22          htmlEntities.put("&Uuml;", "Ü"); htmlEntities.put(" ", " ");
23          htmlEntities.put("&copy;", "\u00a9"); htmlEntities.put("&reg;", "\u00ae");
24          htmlEntities.put("&euro;", "\u20a0"); htmlEntities.put("&rsquo;", "'");
25      }
```

Listing 5: Initialization of *htmlEntities*.

```java
public class QuadruplesCounter {

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "quadruples count");
        job.setJar("QuadruplesCounter.jar");
        job.setJarByClass(QuadruplesCounter.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }

    // Test if s is a valid URI or not
    private static boolean isValidURI(String s) {
        try {
            new URI(s);
            return true;
        } catch (URISyntaxException e) {
            return false;
        }
    }

    public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

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

            String[] values = value.toString().split("\t");
            String subject = values[0];
            String predicate = values[1];
            String object = values[2];
            String provenance = values[3];

            assert isValidURI(subject);
            assert isValidURI(predicate);
            assert isValidURI(provenance);

            if (!isValidURI(object)) { // object is a literal
                word.set(subject);
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

        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 >= 5) {
                result.set(sum);
                context.write(key, result);
            }
        }
    }
}
```

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

```java
public class InOutDegreeCounter {

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

        Job job = Job.getInstance(conf, "in and out degree count");

        job.setJar("InOutDegreeCounter.jar");
        job.setJarByClass(InOutDegreeCounter.class);

        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);

        // Map output
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntTriple.class);

        // Global output
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntTriple.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }

    // Test if s is a valid URI or not
    private static boolean isValidURI(String s) {
        try {
            URI uri = new URI(s);
            return true;
        } catch (URISyntaxException e) {
            return false;
        }
    }
}
```

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

```
public static class TokenizerMapper extends Mapper<Object, Text, Text, IntTriple> {

    private Text word = new Text();

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

        String[] values = value.toString().split("\t");
        String subject = values[0];
        String predicate = values[1];
        String object = values[2];
        String provenance = values[3];

        assert isValidURI(subject);
        assert isValidURI(predicate);
        assert isValidURI(provenance);

        // Structure for the triple is (#Literal,#In,#Out)

        if (isValidURI(object)) { // object is a valid URI
            word.set(subject);
            // increment the out count
            IntTriple triple = new IntTriple(0, 0, 1);
            context.write(word, triple);
        } else { // object is a literal
            word.set(subject);
            // increment the literal count and the out count
            IntTriple triple = new IntTriple(1, 0, 1);
            context.write(word, triple);
        }

        word.set(object);
        // increment the in count for the object
        IntTriple triple2 = new IntTriple(0, 1, 0);
        context.write(word, triple2);
    }
}
```

Listing 8: Mapper class for the last exercise.

```java
public static class IntSumReducer extends Reducer<Text, IntTriple, Text, IntTriple> {
    public void reduce(Text key, Iterable<IntTriple> values, Context context)
            throws IOException, InterruptedException {

        int sumLiteral = 0, sumIn = 0, sumOut = 0;

        for (IntTriple triple : values) {
            sumLiteral += triple.a.get();
            sumIn += triple.b.get();
            sumOut += triple.c.get();
        }

        assert isValidURI(key.toString());

        if (sumLiteral >= 10) {
            context.write(key, new IntTriple(sumLiteral, sumIn, sumOut));
        }
    }
}
```

Listing 9: Reducer class for the last exercise.

```java
static class IntTriple implements Writable {

    private IntWritable a;
    private IntWritable b;
    private IntWritable c;

    public IntTriple() {
        set(new IntWritable(0), new IntWritable(0), new IntWritable(0));
    }

    public IntTriple(IntWritable a, IntWritable b, IntWritable c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public IntTriple(int a, int b, int c) {
        this.a = new IntWritable(a);
        this.b = new IntWritable(b);
        this.c = new IntWritable(c);
    }

    public int compareTo(IntTriple that) {
        int cmp = this.a.compareTo(that.a);
        if (cmp != 0) return cmp;
        cmp = this.b.compareTo(that.b);
        if (cmp != 0) return cmp;
        return this.c.compareTo(that.c);
    }

    public void set(IntWritable a, IntWritable b, IntWritable c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public void write(DataOutput dataOutput) throws IOException {
        a.write(dataOutput);
        b.write(dataOutput);
        c.write(dataOutput);
    }

    public void readFields(DataInput dataInput) throws IOException {
        a.readFields(dataInput);
        b.readFields(dataInput);
        c.readFields(dataInput);
    }

    @Override
    public int hashCode() {
        return a.hashCode() * 163 * 163 + b.hashCode() * 163 + c.hashCode();
    }

    @Override
    public boolean equals(Object obj) {
        if (obj instanceof IntTriple) {
            IntTriple that = (IntTriple) obj;
            return this.a.equals(that.a) &&
                   this.b.equals(that.b) &&
                   this.c.equals(that.c);
        }
        return false;
    }

    @Override
    public String toString() {
        return String.format("(%d,%d,%d)", a.get(), b.get(), c.get());
    }
}
```

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