CURSOS DE VERANO 2014

APROXIMACIÓN PRÁCTICA A LA CIENCIA DE DATOS Y BIG DATA: HERRAMIENTAS KNIME, R, HADOOP Y MAHOUT.

Entorno de Procesamiento Hadoop – Caso Práctico 1

Sara Del Río García

Ejemplo: Word Count

- Tenemos un fichero de gran tamaño que contiene secuencias de palabras.

- **OBJETIVO**: Contar el número de veces que aparece cada palabra en el fichero.

Word Count usando MapReduce

Pseudocódigo:

```
map(key, value):
// key: document ID; value: text of document
    FOR (each word w in value)
        emit(w, 1);

reduce(key, value-list):
// key: a word; value-list: a list of integers (theoretically
// 1, but … "combiners")
        result = 0;
        FOR (each count v on value-list)
            result += v;
        emit(key, result);
```

Word Count usando MapReduce

Main()

```
public static void main(String[] args) throws Exception {
    JobConf conf = new JobConf(WordCount.class);
    conf.setJobName("wordcount");

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

    conf.setMapperClass(Map.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);

    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);

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

    JobClient.runJob(conf);
  }
}
```

Word Count usando MapReduce

Map()

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

  public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
    String line = value.toString();
    StringTokenizer tokenizer = new StringTokenizer(line);
    while (tokenizer.hasMoreTokens()) {
      word.set(tokenizer.nextToken());
      output.collect(word, one);
    }
  }
}
```

Reduce()

```
public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
  public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
    int sum = 0;
    while (values.hasNext()) {
      sum += values.next().get();
    }
    output.collect(key, new IntWritable(sum));
  }
}
```

Word Count usando MapReduce

Uso:

1. **Crear el directorio de entrada "/user/<user>/wordcount/input" en HDFS:**

```
$ hadoop fs -mkdir /user/<user>/wordcount
/user/<user>/wordcount/input
```

Donde <user> es el nombre de usuario Linux del usuario

2. **Crear ficheros de texto de ejemplo y copiarlos al directorio creado previamente en HDFS:**

```
$ echo "Hello World Bye World" > file0
$ echo "Hello Hadoop Goodbye Hadoop" > file1
$ hadoop fs -put file*
/user/<user>/wordcount/input
```

Word Count usando MapReduce

3. **Compilar "WordCount.java":**

```
$ mkdir wordcount_classes
$ javac -cp /usr/lib/hadoop/*:/usr/lib/hadoop-0.20-mapreduce/* -d wordcount_classes WordCount.java
```

4. **Crear el JAR:**

```
$ jar -cvf wordcount.jar -C wordcount_classes/ .
```

5. **Ejecutar la aplicación:**

```
$ hadoop jar wordcount.jar org.myorg.WordCount /user/<user>/wordcount/input /user/<user>/wordcount/output
```

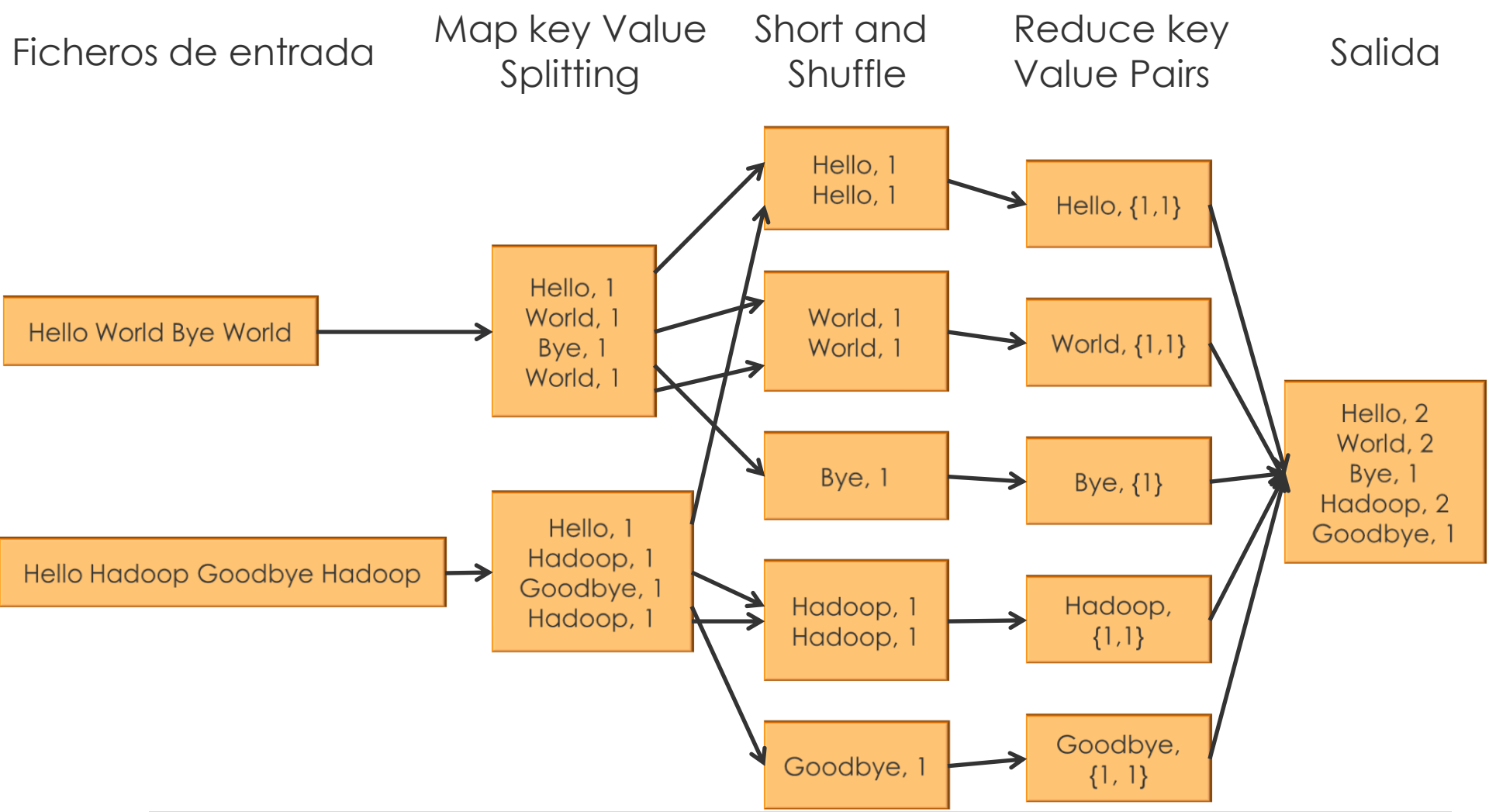
Word Count usando MapReduce

6. **Comprobar la salida:**

```
$ hadoop fs -cat wordcount/output/part-00000

Bye 1
Goodbye 1
Hadoop 2
Hello 2
World 2
```

Word Count usando MapReduce



Referencias

- **Hadoop Tutorial:**

  http://www.cloudera.com/content/cloudera-content/cloudera-docs/HadoopTutorial/CDH4/Hadoop-Tutorial.html