Prácticas BigData

1. Lanzar un proceso MapReduce contra el cluster

- Vamos a usar un fichero de ejemplo de patentes en Estados Unidos, llamado “cite75_99.txt”. Lo tienes disponible en los recursos del capítulo
- Lo subimos a la carpeta prácticas de HDFS

```
hdfs dfs -put acite75_99.txt /practicas
```

- Creamos un programa para trabajar con él. Le vamos a llamar “MyJob.java”. La idea es agrupar las patentes por el código principal
- Ponemos el siguiente contenido

```
import java.io.IOException;
import java.util.Iterator;


import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;


public class MyJob extends Configured implements Tool {


   public static class MapClass extends Mapper<LongWritable, Text, Text, Text> {


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


         String[] citation = value.toString().split(",");
         context.write(new Text(citation[1]), new Text(citation[0]));
      }
```

```
    }

    public static class Reduce extends Reducer<Text, Text, Text, Text> {

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

            String csv = "";
            for (Text val:values) {
                if (csv.length() > 0) csv += ",";
                csv += val.toString();
            }

            context.write(key, new Text(csv));
        }
    }

    public int run(String[] args) throws Exception {
        Configuration conf = getConf();

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

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

        job.setMapperClass(MapClass.class);
        job.setReducerClass(Reduce.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        System.exit(job.waitForCompletion(true)?0:1);

        return 0;
    }
```

```
    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new MyJob(), args);

        System.exit(res);
    }
}
```

- Exportamos la librería para localizarla en el CLASSPATH

```
export HADOOP_CLASSPATH=$JAVA_HOME/lib/tools.jar
```

- Compilamos

```
hadoop com.sun.tools.javac.Main MyJob.java
```

- Creamos el JAR correspondiente

```
jar cvf MyJob.jar My*
manifiesto agregado
agregando: MyJob.class(entrada = 2028) (salida = 967)(desinflado 52%)
agregando: Myjob.java(entrada = 2657) (salida = 739)(desinflado 72%)
agregando: MyJob$MapClass.class(entrada = 1490) (salida = 585)(desinflado 60%)
agregando: MyJob$Reduce.class(entrada = 1811) (salida = 786)(desinflado 56%)
```

- Ejecutamos

```
hadoop jar MyJob.jar MyJob /practicas/cite75_99.txt /resultado7
```

- Comprobamos el resultado dejado en el directorio “resultado7”

```
hdfs dfs -cat /resultado7/part-r-00000
```

- Podemos ver en la página de Admon de YARN donde se ha ejecutado cada uno de los procesos, mapper y reducer en su caso