Conceptos y Aplicaciones de Big Data

MapReduce

Desarrollo en Java y en Python

Prof. Waldo Hasperué
whasperue@lidi.info.unlp.edu.ar

Temario

- Desarrollo de aplicaciones en MapReduce
  - Java
  - Python

Ejemplo - WordCount

- WordCount es un programa que contabiliza la ocurrencia de cada palabra que aparece en un texto.
- Ejemplo:
  - Entrada:

    "Si tú crees que puedes, puedes. Si tú crees que no puedes, no puedes"
  - Salida:

Puedes	4	Crees	2
Si	2	Que	2
Tú	2	No	2

MAPREDUCE

- Desarrollo de una aplicación en Java

Apache MapReduce API

- Mapper: superclase de toda tarea map
- Reducer: superclase para toda tarea reducer
- Job: Representa un trabajo map-reduce
- Tool: Representa al JobTracker
- TextInputFormat: Tipo de dato (texto) en el input <LongWritable, Text>
- TextOutputFormat: Tipo de dato (texto) en el output

Apache MapReduce API

- Definidos en la API
  - ArrayWritable
  - BooleanWritable
  - ByteWritable
  - DoubleWritable
  - FloatWritable
  - IntWritable
  - LongWritable
  - ShortWritable
  - Text
- Definidos por el usuario
  - Implementar WritableComparable para las claves
  - Implementar Writable para los valores

WordCount - Main

```
package miPrimerProgramaEnHadoop;

import org.apache.hadoop.util.ToolRunner;

public class Main {

    public static void main(String[] args) throws Exception{

        System.exit(ToolRunner.run(null, new Worker(), args));

    }
}
```

Punto de partida para la ejecución de cualquier proceso en Hadoop.
Worker es la clase que implementaremos para resolver los diferentes problemas

WordCount - Mapper

```
public class WCMapper extends Mapper<LongWritable, Text, Text, LongWritable>{

    private static LongWritable one = new LongWritable(1);

    public void map(LongWritable key, Text value, Context context) throws
                                        IOException, InterruptedException{
            Text word = new Text();
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line, "\"().,[]/-' ;", false);

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

WordCount - Mapper

```
public class WCMapper extends Mapper<LongWritable, Text, Text, LongWritable>{

    private static LongWritable one = new LongWritable(1);

    public void map(LongWritable key, Text value, Context context) throws
                                        IOException, InterruptedException{
            Text word = new Text();
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line, "\"().,[]/-' ;", false);

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

Toda tarea mapper tiene que heredar de *Mapper* e implementar el método *map*.

WordCount - Mapper

```
public class WCMapper extends Mapper<LongWritable, Text, Text, LongWritable>{

    private static LongWritable one = new LongWritable(1);

    public void map(LongWritable key, Text value, Context context) throws
                                        IOException, InterruptedException{
            Text word = new Text();
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line, "\"().,[]/-' ;", false);

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

Definir los tipos de clave-valor de entrada y salida de la tarea *Map* (k1, v1, k2, v2)

WordCount - Mapper

```
public class WCMapper extends Mapper<LongWritable, Text, Text, LongWritable>{

    private static LongWritable one = new LongWritable(1);

    public void map(LongWritable key, Text value, Context context) throws
                                        IOException, InterruptedException{
            Text word = new Text();
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line, "\"().,[]/-' ;", false);

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

Debe coincidir con los tipos de clave-valor de entrada (k1, v1)

WordCount - Mapper

```
public class WCMapper extends Mapper<LongWritable, Text, Text, LongWritable>{

    private static LongWritable one = new LongWritable(1);

    public void map(LongWritable key, Text value, Context context) throws
                                        IOException, InterruptedException{
            Text word = new Text();
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line, "\"().,[]/-' ;", false);

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

Esta es la escritura, debe coincidir con los tipos de clave - valor de salida (k2, v2)

WordCount - Reducer

```
public class WCReducer extends Reducer<Text, LongWritable, Text, LongWritable> {

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

            int times = 0;
            for (@SuppressWarnings("unused") Object val : values) {
                times++;
            }

            context.write(key, new LongWritable(times));
    }
}
```

WordCount - Reducer

```
public class WCReducer extends Reducer<Text, LongWritable, Text, LongWritable> {

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

            int times = 0;
            for (@SuppressWarnings("unused") Object val : values) {
                times++;
            }

            context.write(key, new LongWritable(times));
    }
}
```

Toda tarea reducer tiene que heredar de *Reducer* e implementar el método *reduce*

WordCount - Reducer

```
public class WCReducer extends Reducer<Text, LongWritable, Text, LongWritable> {

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

            int times = 0;
            for (@SuppressWarnings("unused") Object val : values) {
                times++;
            }

            context.write(key, new LongWritable(times));
    }
}
```

Definir los tipos de clave-valor de entrada y salida de la tarea reduce.
La entrada debería coincidir con la salida de la tarea map (k2, v2, k3, v3)

WordCount - Reducer

```
public class WCReducer extends Reducer<Text, LongWritable, Text, LongWritable> {

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

        int times = 0;
        for (@SuppressWarnings("unused") Object val : values) {
            times++;
        }

        context.write(key, new LongWritable(times));
    }
}
```

Debe coincidir con los tipos de clave-valor de entrada (k2, v2).
Notar que como valor en realidad se recibe una lista de valores (interface Iterable)

WordCount - Reducer

```
public class WCReducer extends Reducer<Text, LongWritable, Text, LongWritable> {

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

            int times = 0;
            for (@SuppressWarnings("unused") Object val : values) {
                times++;
            }

            context.write(key, new LongWritable(times));
    }
}
```

Debe coincidir con los tipos de clave-valor de salida (k3, v3).

WordCount - Driver

```
public class Worker extends Configured implements Tool {
        …
        @Override
        public int run(String[] args) throws Exception {
            Job job;
            boolean success;

            job = setupJob();
            success = job.waitForCompletion(true);
            if (!success){
                    System.out.println("Error job");
                    return -1;
            }

            return 0;
        }
}
```

WordCount - Driver

```
public class Worker extends Configured implements Tool {
        …
        @Override
        public int run(String[] args                    on {
            Job job;
            boolean succ

            job = setupJob();
            success = job.waitForCompletion(true);
            if (!success){
                    System.out.println("Error job");
                    return -1;
            }

            return 0;
        }
}
```

Nuestro *Worker* debe heredar de *Configured* e implementar la interface *Tool*.

WordCount - Driver

```
public class Worker extends Configured implements Tool {
    ...
    @Override
    public int run(String[] args) throws Exception {
        Job job;
        boolean success;

        job = setu
        success =                    Completion(true);

                                     or job");

        }

        return 0;
    }
}
```

Debe re-implementar el método *run*.

WordCount - Driver

```
public class Worker extends Configured implements Tool {
        …
        @Override
        public int run(String[] args) throws Exception {
            Job job;
            boolean success
            job = setupJo
            success = j                ompletion(true);
                                                    or job");
            }

            return 0;
        }
}
```

Recibimos los argumentos pasados por línea de comando

MapReduce - Driver

```
public class Worker extends Configured implements Tool {
        …
        @Override
        public int run(String[] args) throws Exception {
            Job job;
            boolean success;

            job = setupJob();
            success = job.waitForCompletion(true);
            if (!success){
                    System.out.println("Error job");
                    return -1;
            }

            return 0;
        }
}
```

Se prepara el *Job* (*setupJob*) y luego se lo ejecuta (*waitForCompletion*)

MapReduce - Driver

```
public class Worker extends Configured implements Tool {
    private Job setupJob() throws IOException{
            Configuration conf = getConf();
            Job job = new Job(conf, "WordCount");
            job.setJarByClass(Worker.class);

            //configure Mapper
            job.setMapperClass(WCMapper.class);
            job.setMapOutputKeyClass(Text.class);
            job.setMapOutputValueClass(LongWritable.class);

            //configure Reducer
            job.setReducerClass(WCReducer.class);
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(LongWritable.class);

             …
```

MapReduce - Driver

```
public class Worker extends Configured implements Tool {
    private Job setupJob() throws IOException{
        Configuration conf = getConf();
        Job job = new Job(conf, "WordCount");
        job.setJarByClass(Worker.class);

        //configure Mapper
        job.setMapperClass(WCMapper.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(LongWritable.class);

        //configure Reducer
        job.setReducerClass(WCReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(LongWritable.class);

        …
```

Se crea el job

MapReduce - Driver

```
public class Worker extends Configured implements Tool {
    private Job setupJob() throws IOException{
            Configuration conf = getConf();
            Job job = new Job(conf, "WordCount");
            job.setJarByClass(Worker.class);

            //configure Mapper
            job.setMapperClass(WCMapper.class);
            job.setMapOutputKeyClass(Text.class);
            job.setMapOutputValueClass(LongWritable.class);

            //configure Reducer
            job.setReducerClass(WCReducer.class);
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(LongWritable.class);

             …
```

Se configura el mapper estableciendo la subclase de *Map* a ejecutar y los tipos clave valor de salida (k2, v2).

MapReduce - Driver

```
public class Worker extends Configured implements Tool {
    private Job setupJob() throws IOException{
            Configuration conf = getConf();
            Job job = new Job(conf, "WordCount");
            job.setJarByClass(Worker.class);

            //configure Mapper
            job.setMapperClass(WCMapper.class);
            job.setMapOutputKeyClass(Text.class);
            job.setMapOutputValueClass(LongWritable.class);

            //configure Reducer
            job.setReducerClass(WCReducer.class);
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(LongWritable.class);

             …
```

Se configura el reducer estableciendo la subclase de *Reduce* a ejecutar y los tipos clave valor de salida (k3, v3).

MapReduce – Driver

```
public class Worker extends Configured implements Tool {
    private Job setupJob() throws IOException{
            …
            job.setNumReduceTasks(3);
            job.setInputFormatClass(TextInputFormat.class);
            job.setOutputFormatClass(TextOutputFormat.class);

            FileInputFormat.addInputPath(job, new Path("input"));
            FileOutputFormat.setOutputPath(job, new Path("output"));

            return job;
    }
```

MapReduce – Driver

```
public class Worker extends Configured implements Tool {
    private Job setupJob() throws IOException{
        Job job = new Job(getConf());
        job.setNumReduceTasks(3);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job, new Path("input"));
        FileOutputFormat.setOutputPath(job, new Path("output"));

        return job;
    }
```

Se establece el tipo de <clave, valor> tanto en la entrada del proceso como en la salida
Opcionalmente es posible definir el número de tareas reduce que se deben ejecutar.

MapReduce – Driver

```
public class Worker extends Configured implements Tool {
    private Job setupJob() throws IOException{
            …
            job.setNumReduceTasks(3);
            job.setInputFormatClass(TextInputFormat.class);
            job.setOutputFormatClass(TextOutputFormat.class);

            FileInputFormat.addInputPath(job, new Path("input"));
            FileOutputFormat.setOutputPath(job, new Path("output"));

            return job;
    }
```

Se establecen los path dentro del HDFS donde estarán los archivos de entrada y donde quedarán los archivos de salida.

Compilando un proyecto

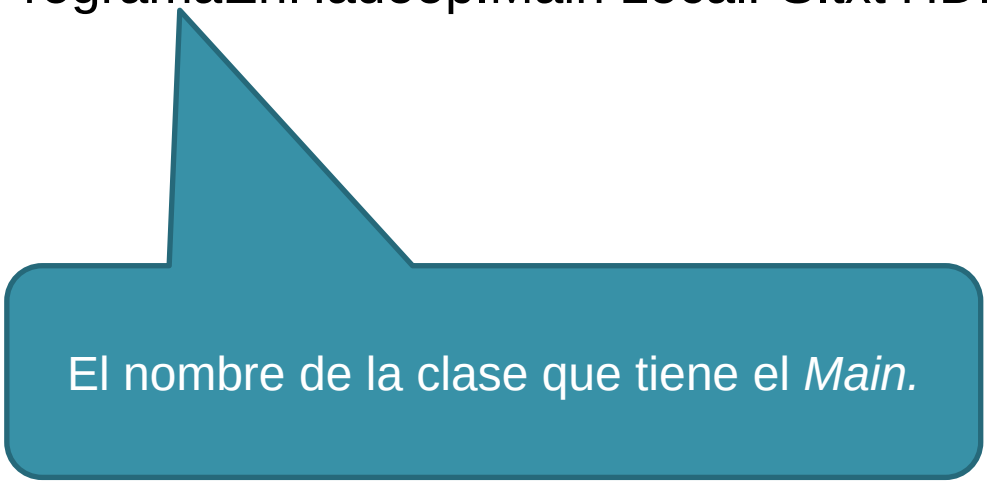
- Estructura de directorios
  - src
  - bin
  - input
  - output

Compilando un proyecto

- Compilando en VirtualBox
  - javac **src**/*.java -classpath /home/hduser/jarHadoop/commons-cli-1.2.jar:/home/hduser/jarHadoop/hadoop-common-2.6.0.jar:/home/hduser/jarHadoop/hadoop-mapreduce-client-core-2.6.0.jar -d bin/
- Compilando en Docker
  - javac **src**/*.java -classpath HADOOP_HOME/commons-cli-1.2.jar:HADOOP_HOME/hadoop-common-3.1.2.jar:HADOOP_HOME/hadoop-mapreduce-client-core-3.1.2.jar -d bin/
- Empaquetando
  - jar -cvf ./**bin**/miPrimerProgramaEnHadoop.jar -C ./**bin**/ .

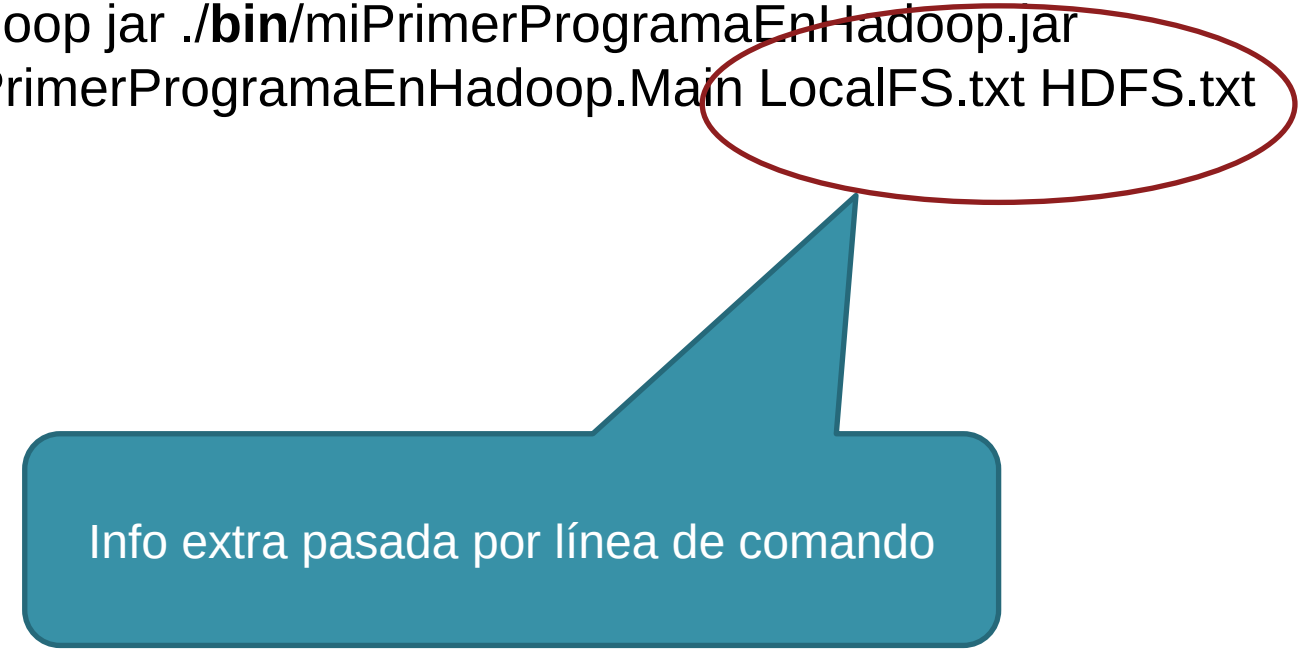
Ejecutando un proyecto

- Ejecutando
  - hadoop jar ./**bin**/miPrimerProgramaEnHadoop.jar miPrimerProgramaEnHadoop.Main LocalFS.txt HDFS.txt



Ejecutando un proyecto

- Ejecutando
  - hadoop jar ./**bin**/miPrimerProgramaEnHadoop.jar miPrimerProgramaEnHadoop.Main LocalFS.txt HDFS.txt



MAPREDUCE

- Desarrollo de una aplicación en Python

Aplicación en Python

- La ejecución de una aplicación en Python se realiza mediante una API de hadoop llamada "hadoop-streaming" que lee el contenido del DFS y lo envía como dato a la entrada estándar (STDIN) para ser consumida por la aplicación en Python.
- Lo escrito por la aplicación en la salida estándar (STDOUT) se almacena en el DFS.
- Una aplicación en Python no tiene un proceso driver, solo hay que escribir un script para el proceso mapper y otro para el reducer.

WordCount - Mapper

```
for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print '%s\t%s' % (word, 1)
```

WordCount - Mapper

```
for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print '%s\t%s' % (word, 1)
```

La entrada se recibe desde STDIN

WordCount - Mapper

```
for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print '%s\t%s' % (word, 1)
```

La salida debe imprimirse con formato clave-valor en STDOUT

WordCount - Mapper

```
for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print '%s\t%s' % (word, 1)
```

La salida debe imprimirse con formato clave-valor en STDOUT

WordCount - Reducer

```
current_word = None
current_count = 0
word = None
for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    count = int(count)

    if current_word == word:
        current_count += count
    else:
        if current_word != None:
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word

if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

WordCount - Reducer

```
current_word = None
current_count = 0
word = None
for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    count = int(cou

    if current_word ==
        current_count +=
    else:
        if current_word != No
            print '%s\t%s' % (c                d, current_count)
        current_count = c
        current_word = wo

if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

La entrada se recibe desde STDIN

WordCount - Reducer

```
current_word = None
current_count = 0
word = None
for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    count = int(count)

    if current_word == word:
        current_count += coun
    else:
        if current_word != None
            print '%s\t%s' % (cu                d, current_count)
        current_count = c
        current_word = wo

if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

Se recupera la clave-valor

WordCount - Reducer

```
current_word = None
current_count = 0
word = None
for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    count = int(count)

    if current_word == word:
        current_count += count
    else:
        if current_word != N    e:
            print '%s\t%s' % (c          word, current_count)
        current_count = co
        current_word = wo

if current_word == word:
    print '%s\t%s' % (cur
```

Si bien en la entrada se puede recibir (de manera ordenada) más de una clave, el control de cambio de clave se debe hacer manualmente

WordCount - Reducer

```
current_word = None
current_count = 0
word = None
for line in sys.stdin:
    line = line.strip()
    word, count = line.sp
    count = int(count)

    if current_word == word:
        current_count += count
    else:
        if current_word != None:
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word

if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

La salida debe enviarse con formato clave-valor al STDOUT

WordCount - Ejecución

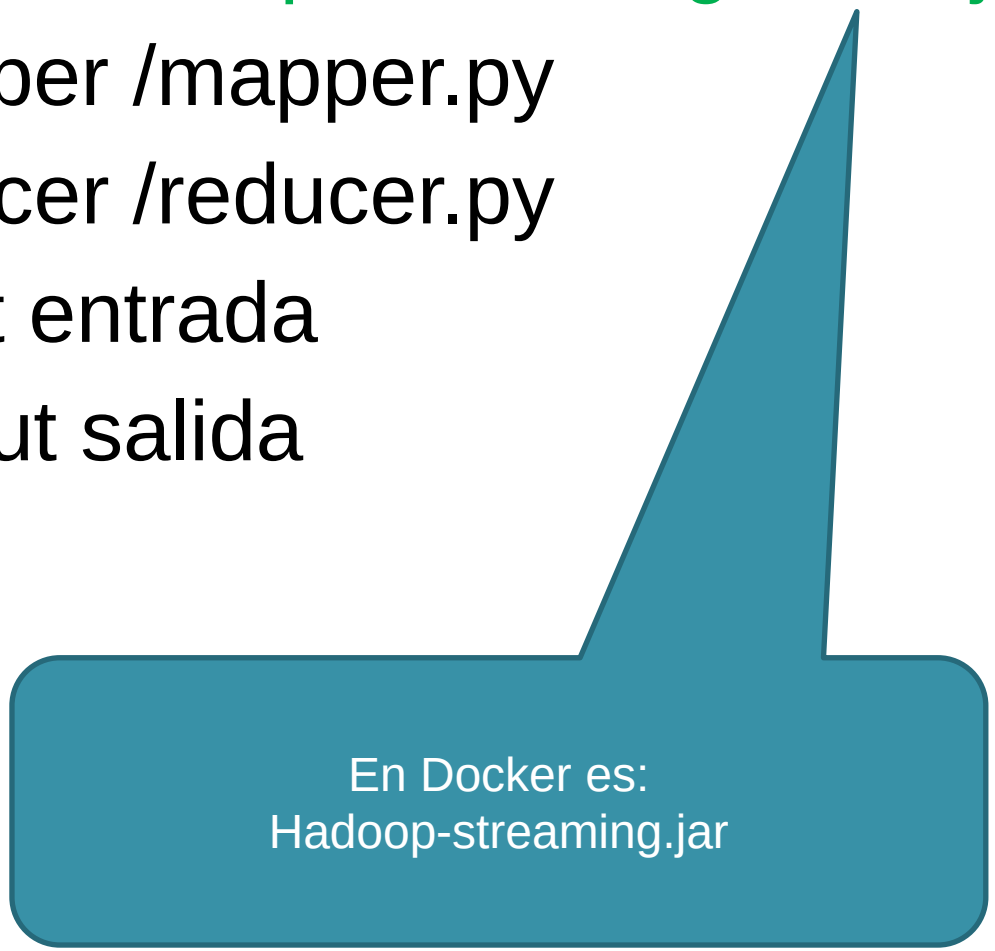
hadoop jar hadoop-streaming-2.6.0.jar

-mapper /mapper.py

-reducer /reducer.py

-input entrada

-output salida



WordCount - Ejecución

```
hadoop jar hadoop-streaming-2.6.0.jar
    -mapper /mapper.py
    -reducer /reducer.py
    -input entrada
    -output salida
```

Los archivos con el mapper y el reducer deben estar con su ruta absoluta al FS local

WordCount - Ejecución

```
hadoop jar hadoop-streaming-2.6.0.jar
        -mapper /mapper.py
        -reducer /reducer.py
        -input entrada
        -output salida
```

Los directorios de entrada y salida son directorios en el DFS. El directorio de salida NO debe existir, si existe la ejecución falla.