COMO USAR HADOOP

Y sobrevivir a la experiencia

ORGANIZACIÓN

- Descripción Hadoop:
  - Procesos involucrados
  - Esquema de Funcionamiento
- Instalación de Hadoop
  - Instalación Local
  - Descripción de Instalación en Cluster

ORGANIZACIÓN

- HDFS:
  - Acceder al sistema de ficheros de Hadoop.
  - Carga y descarga de Información
- Ejecución de Procesos
  - Lanzamiento, ejecución y verificación de procesos (en local)
  - Lanzamiento, ejecución y verificacion de procesos (cluster)

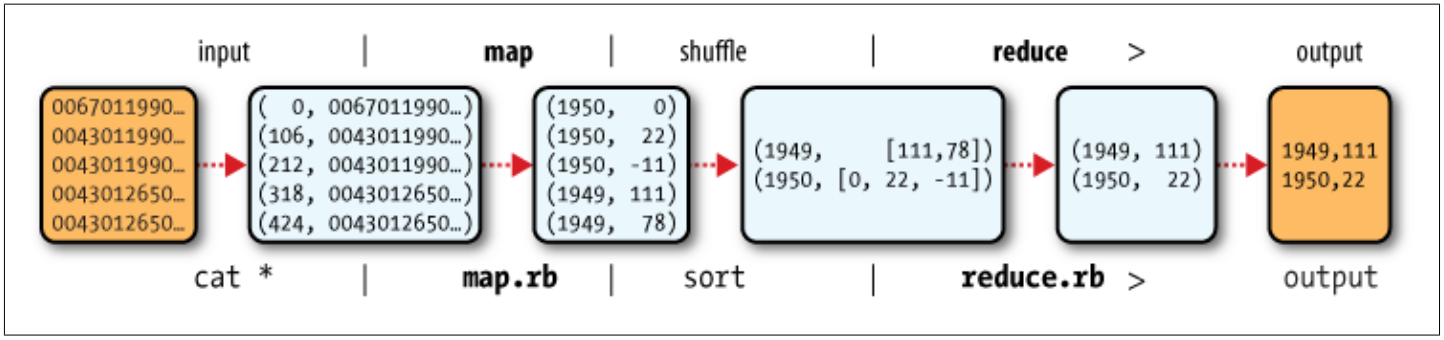
REPOSITORIO DE INFORMACÓN

- https://www.tsc.uc3m.es/~hmolina/mlgp
  - Acceso restringido con password del Departamento

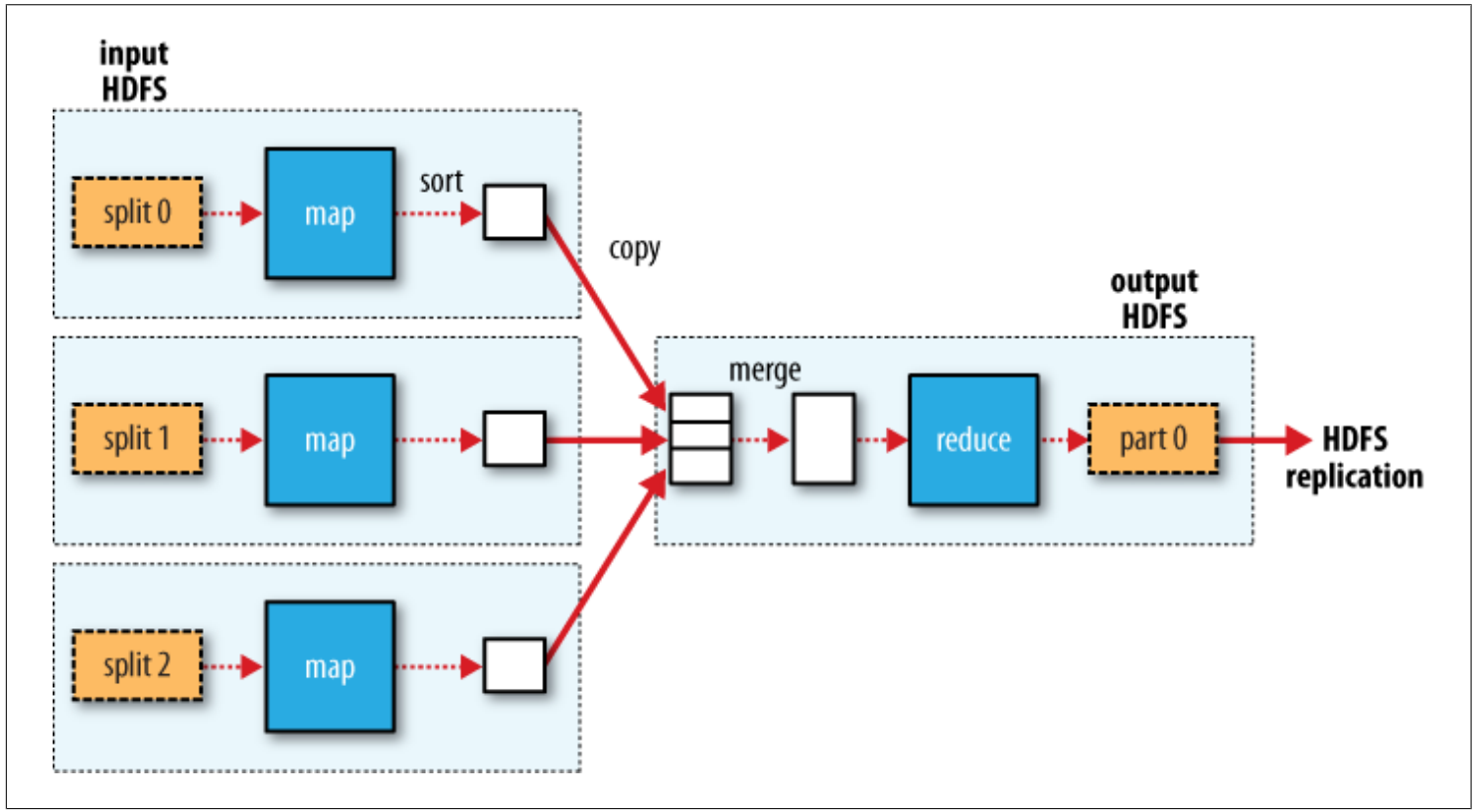
HADOOP

Instalación, configuración y uso

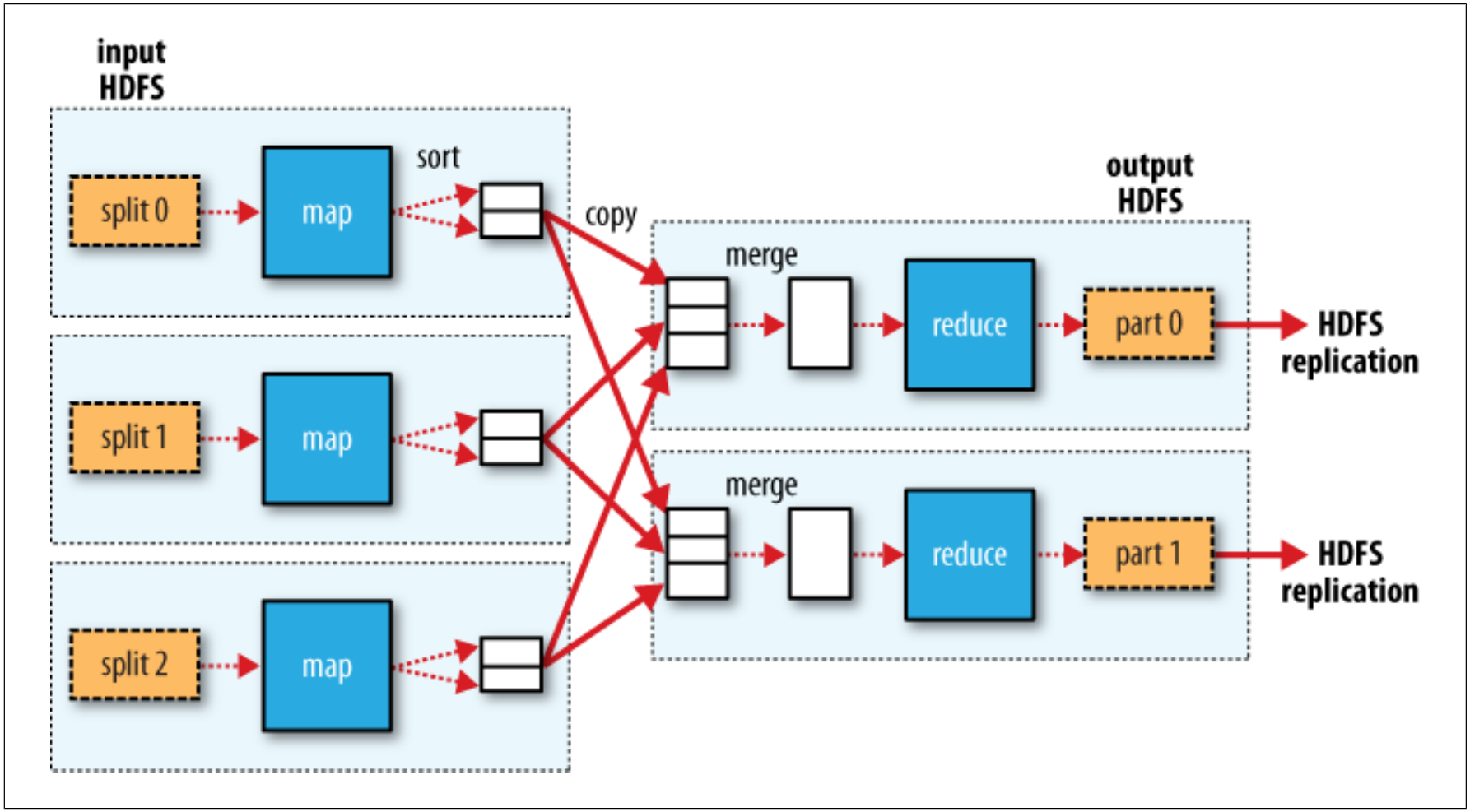
FLUJO DE DATOS DE HADOOP



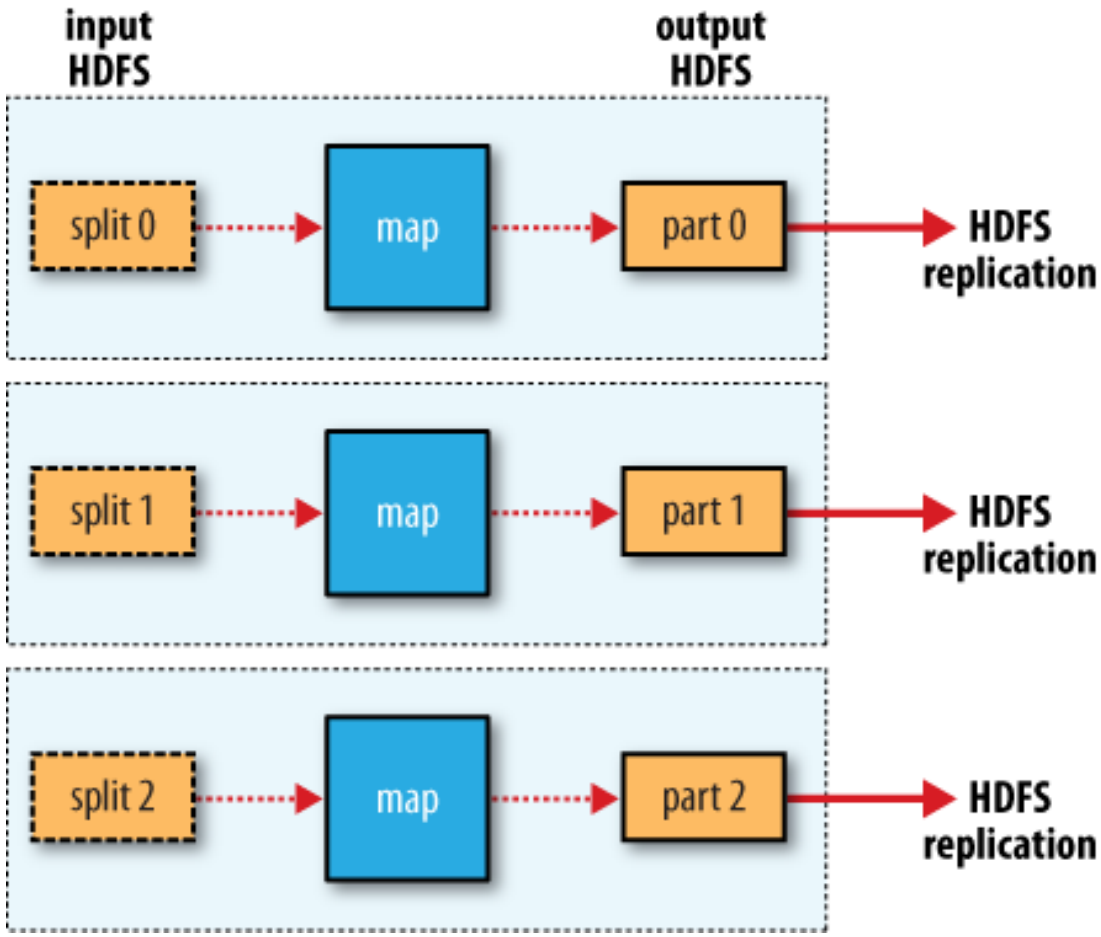
ARQUITECTURA TÍPICA



ARQUITECTURA MÚLTIPLES REDUCTORES



ARQUITECTURA SIN REDUCTORES



HADOOP

- Varias formas de ejecución:
  - En modo Standalone: No se necesita configurar nada.
  - En modo Servidor - nodo local: Un sistema basado en cliente servidor, pero que se ejecuta en modo local todo.
  - En modo distribuido: Infraestructura completa con varios nodos de almacenamiento, ejecución, etc...

MODO STANDALONE

- Descomprimir la distribución de Hadoop
- Establecer variable JAVA_HOME
- Et Voilà!!!!

PRUEBA

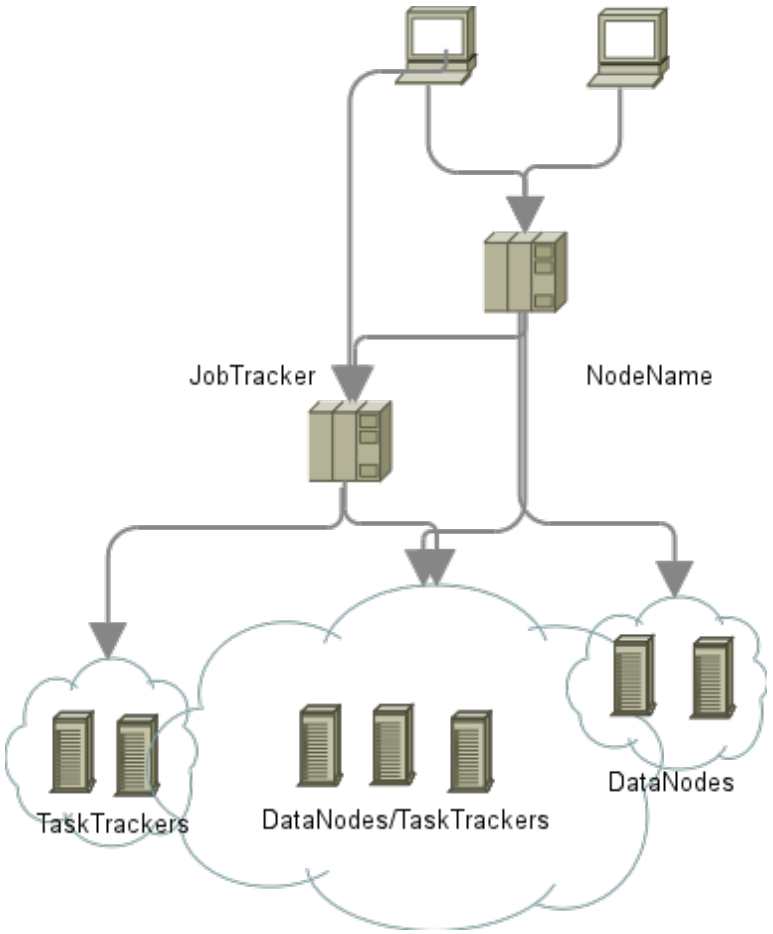
- Descomprimir BBDD de Reuters
- Ejecutar el comando:

```
hadoop jar hadoop-examples-1.1.2.jar

hadoop jar hadoop-examples-1.1.2.jar wordcount dir_reuters
dir_output
```

- El directorio dir_output no debe existir
- Observar demora

ESTRUCTURA DE HADOOP



CONFIGURACIÓN EN MODO SERVIDOR LOCAL

- Creamos un directorio llamado
    - conf_single
- Copiamos los contenidos de conf a conf_single

CONFIGURACIÓN DEL SERVIDOR MAESTRO CORE-SITE.XML

- Define el servidor que contendrá el sistema de ficheros

```
<configuration>
     <property>
         <name>fs.default.name</name>
         <value>hdfs://localhost:9000</value>
     </property>
     <property>
         <name>hadoop.tmp.dir</name>
         <value>/tmp/hadoop/tmp</value>
     </property>
</configuration>
```

HDFS-SITE.XML

- Define la configuración del comportamiento del sistema distribuido de ficheros

```
<configuration>
   <property>
      <name>dfs.name.dir</name>
      <value>/tmp/hadoop/name</value>
   </property>
   <property>
      <name>dfs.replication</name>
      <value>1</value>
   </property>
   <property>
      <name>dfs.data.dir</name>
      <value>/tmp/hadoop/data</value>
   </property>
</configuration>
```

En instalaciones standalone se configura que la información no esté replicada. En configuraciones en cluster, la información DEBE estar replicada

CONFIGURACIÓN DEL JOBTRACKER MAPRED-SITE.XML

- Configura el coordinador de tareas

```
<configuration>
   <property>
      <name>mapred.job.tracker</name>
      <value>localhost:9001</value>
   </property>
   <property>
      <name>mapred.system.dir</name>
      <value>/hadoop/mapred/system</value>
   </property>
   <property>
      <name>mapred.local.dir</name>
      <value>/tmp/hadoop/tmp_mapred</value>
   </property>
</configuration>
```

OTROS FICHEROS

- Hay que editar el fichero hadoop-env.sh

```
# The java implementation to use.  Required.
export JAVA_HOME="/Library/Java/JavaVirtualMachines/
jdk1.7.0_21.jdk/Contents/Home"

export HADOOP_OPTS="-Djava.net.preferIPv4Stack=true -server"
```

INICIALIZACION DEL DFS

- Ejecutar:

```
hadoop --config conf_single namenode -format
hadoop-daemon.sh --config conf_single start namenode
hadoop --config conf_single dfs -mkdir /user
hadoop --config conf_single dfs -chmod 755 /user
hadoop --config conf_single dfs -mkdir /tmp
hadoop --config conf_single dfs -chmod 777 /tmp
hadoop --config conf_single dfs -mkdir /mapred
hadoop --config conf_single dfs -chmod 755 /mapred
```

INICIAR EL SISTEMA

- bin/start-all.sh --config conf_single
- Acceso al estado de los sistemas:
  - NameNode (DFS) http://localhost:50070
  - JobTracker: http://localhost:50030

ACCESO AL SISTEMA DE FICHEROS

- El comando bin/hadoop invoca la API básica de hadoop.
- La aplicación dfs de la API básica permite el acceso al sistema de ficheros.
  - bin/hadoop --config conf_single dfs

DFS

- Basado en los comandos UNIX
  - hadoop --config conf_single dfs -ls
  - hadoop --config conf_single dfs -mkdir
  - hadoop --config conf_single dfs -chown
  - hadoop --config conf_single dfs -chmod

DFS

- Para subir ficheros del ordenador local al DFS
  - hadoop --config conf_single dfs -put src dst
  - hadoop --config conf_single dfs -copyFromLocal src dst
- Para descargar ficheros
  - hadoop --config conf_single dfs -get src dst
  - hadoop --config conf_single dfs -copyToLocal

INVOCAR UNA APLICACION

- Si está en un fichero jar:
  - hadoop jar FICHERO_JAR.jar ClaseMain [parametros]

PRIMERA PRUEBA: CONTAR PALABRAS

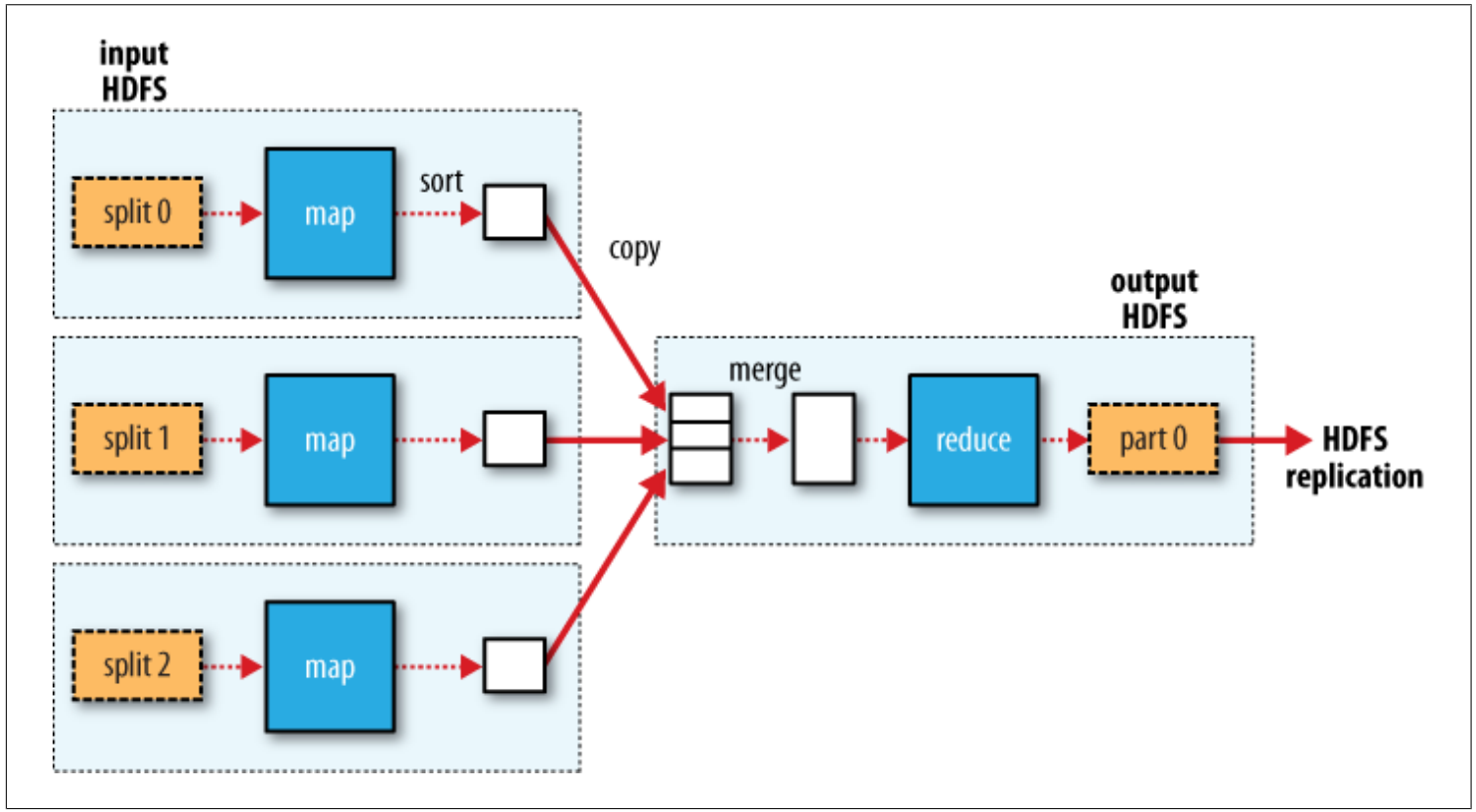
- Se utilizará el programas ejemplos proporcionados por hadoop:
  - hadoop --config conf_cluster jar hadoop-examples-1.1.2.jar
- En particular wordcount
  - hadoop --config conf_cluster jar hadoop-examples-1.1.2.jar wordcount

INTEGRACIÓN DE HADOOP Y PYTHON

MAPPER -> REDUCER

- Los datos se presentan por tuplas:
    - <llave><dato>
- y se deben presentar
    - <llave><dato>

ARQUITECTURA TÍPICA



CALCULO DE MAXIMA TEMPERATURA ANUAL

- Base de datos de sensores en E.E.U.U.
- Datos no ordenados tipo texto
- Estructura simple
  - Datos de interes:
    - Año: cols 15 a 18
    - Temperatura: cols 104 a 106. Si vale 9999 no es lectura válida
    - Col 137 debe valer 0 o 1

```
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MaxTemperatureMapper
  extends Mapper<LongWritable, Text, Text, IntWritable> {

        private static final int MISSING = 999;

        @Override
        public void map (LongWritable key, Text value, Context context)
          throws IOException, InterruptedException {
                String line = value.toString();
                String year = line.substring(14,18);
                int airTemperature;
                String quality = line.substring(136,137);
                if (quality.matches("[01]") ) {
              airTemperature =
Integer.parseInt(line.substring(103,106).trim());

                  if (airTemperature != MISSING )
                        context.write(new Text(year), new
IntWritable(airTemperature));
                }

        }

}
```

```
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MaxTemperatureReducer
  extends Reducer<Text, IntWritable, Text, IntWritable> {

        @Override
        public void reduce(Text key, Iterable<IntWritable> values,
Context context)
          throws IOException, InterruptedException{
                int maxValue = Integer.MIN_VALUE;
                for (IntWritable value : values ) {
                  maxValue = Math.max(maxValue,  value.get());
                }
                context.write(key, new IntWritable(maxValue));
        }
```

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;



public class MaxTemperature {

        /**
         * @param args
         */
        public static void main(String[] args) throws Exception{
                if (args.length != 2) {
                        System.err.println("Usage: MaxTemperature <input
path> <output path>");
                        System.exit(-1);
                }

                Job job = new Job();
                job.setJarByClass(MaxTemperature.class);
                job.setJobName("Max Temperature");

                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));

                job.setMapperClass(MaxTemperatureMapper.class);
                job.setReducerClass(MaxTemperatureReducer.class);

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

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

}
```

EJEMPLO MAX TEMPERATURE

- Ejemplo básico de MaxTem

```
hadoop jar MaxTemp.jar MaxTemperature\
 <dfs_src> \
 <dfs_dst>
```

TÉCNICAS

- Streaming:
  - Las aplicaciones para hacer MAP y REDUCE se escriben en un lenguaje que permita leer información desde STDIN, y escribir a STDOUT
  - Se invoca una aplicación Hadoop que distribuye un proceso a los nodos de cómputo, el cual invoca la aplicación de MAP/REDUCE que se le ha indicado

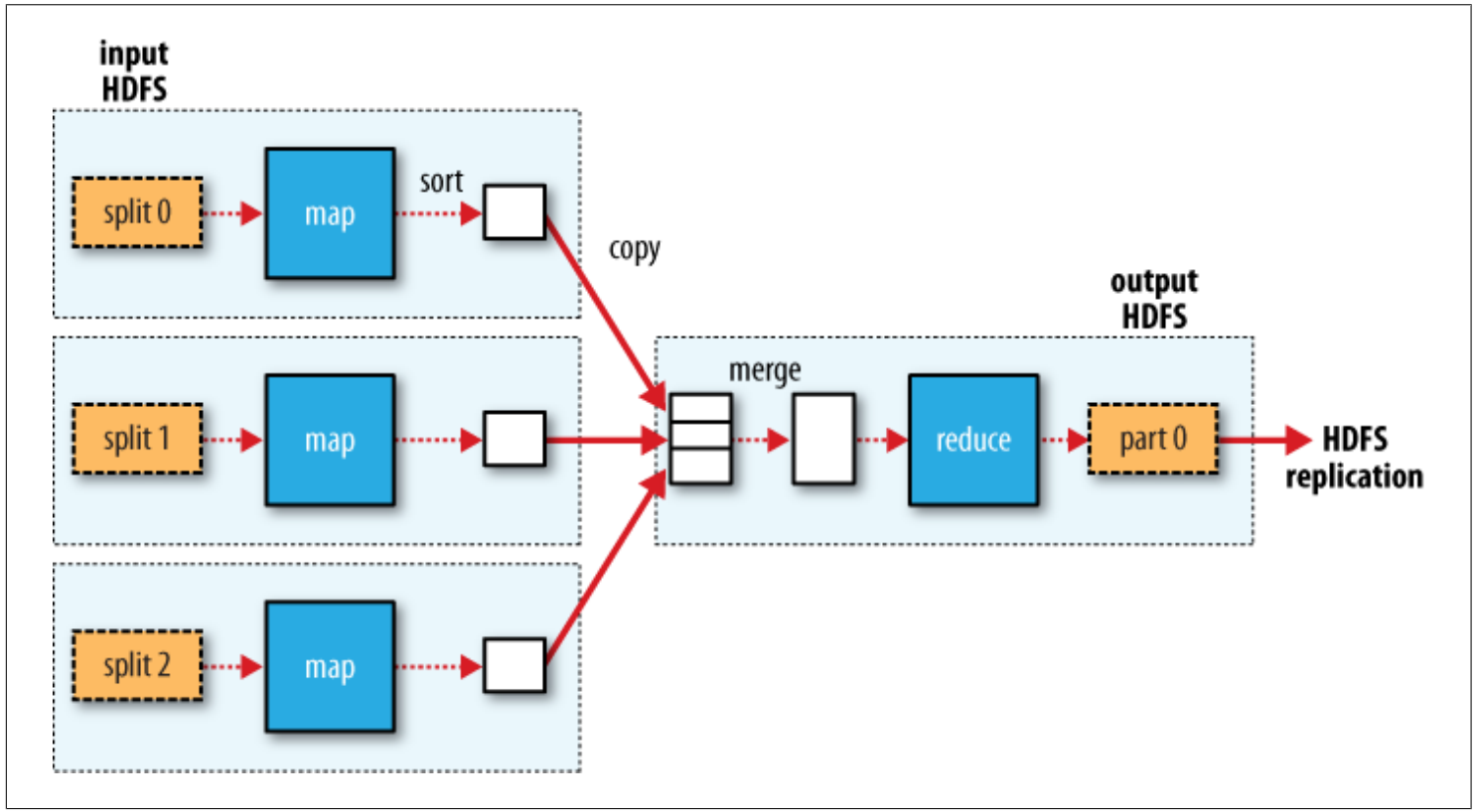
- hadoop jar ../../hadoop-1.1.2/contrib/streaming/hadoop-streaming-1.1.2.jar

```
hadoop jar \
  contrib/streaming/hadoop-streaming-1.1.2.jar \
 -input {DFSDIR}/input -output {DFSDIR}/output_py \
 -mapper {GLOBAL_DIR}/map.py \
 -reducer {GLOBAL_DIR}/reduce.py
```

STREAMING (CONT)

- La aplicación que hace MAP/REDUCE DEBE SER ACCESIBLE en el sistema de ficheros normal de cada nodo (no en el DFS de Hadoop)
  - Solución: Copiar los ficheros a clusterdata

ARQUITECTURA COMBINER



```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;



public class MaxTemperature {

        /**
         * @param args
         */
        public static void main(String[] args) throws Exception{
                if (args.length != 2) {
                        System.err.println("Usage: MaxTemperature <input path> <output
path>");
                        System.exit(-1);
                }

                Job job = new Job();
                job.setJarByClass(MaxTemperature.class);
                job.setJobName("Max Temperature");

                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));

                job.setMapperClass(MaxTemperatureMapper.class);
                job.setCombinerClass(MaxTemperatureReducer.class);
                job.setReducerClass(MaxTemperatureReducer.class);

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

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

}
```

CON COMBINER

- hadoop jar ../../hadoop-1.1.2/contrib/streaming/hadoop-streaming-1.1.2.jar

```
hadoop jar hadoop-streaming-1.1.2.jar \
 -input {DFSDIR}/input -output {DFSDIR}/output_py \
 -mapper {GLOBAL_DIR}/map.py \
 -combiner {GLOBAL_DIR}/reduce.py
 -reducer {GLOBAL_DIR}/reduce.py
```

http://nostromo.tsc.uc3m.es:50030/jobtracker.jsp

http://subserver1.tsc.uc3m.es:50070/dfshealth.jsp

BIBILIOGRAFIA

- Hadoop: The Definitive Guide. 3rd Edition. Tom White. O'Reilly

- Writing an Hadoop MapReduce Program in Python

- Hadoop 1.1.2 Documentation